AN

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06T 15/50 | A1

(11) International Publication Number: WO 99/16021

(43) International Publication Date: 1 April 1999 (01.04.99)

(21) International Application Number: PCT/US98/20096

(22) International Filing Date: 25 September 1998 (25.09.98)

(30) Priority Data:
08/937,793 25 September 1997 (25.09.97) US
09/070,809 1 May 1998 (01.05.98) US

(71) Applicant: SILICON GRAPHICS, INC. [US/US]; 2011 North Shoreline Boulevard, Mountain View, CA 94043-1389 (US).

(72) Inventors: BAUM, Daniel, R.; 1181 Lincoln Avenue, Palo Alto, CA 94301 (US). HANRAHAN, Patrick, M.; 40 Minoca Road, Portola Valley, CA 94028 (US). PHARR, Matthew, M.; Apartment A, 285 College Avenue, Palo Alto, CA 94306 (US).

(74) Agents: RAY, Michael, B. et al.; Sterne, Kessler, Goldstein & Fox P.L.L.C., Suite 600, 1100 New York Avenue, N.W., Washington, DC 20005-3934 (US).

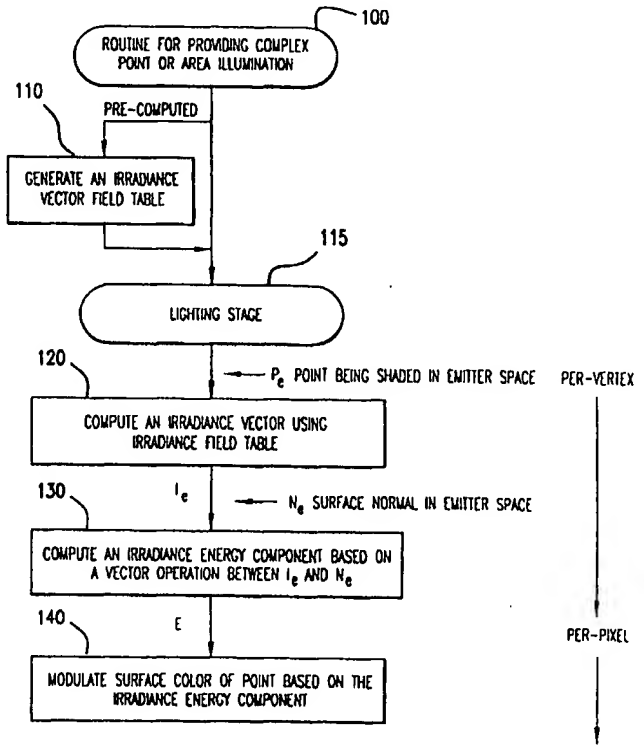
(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR PROVIDING ILLUMINATION IN COMPUTER GRAPHICS SHADING AND ANIMATION

(57) Abstract

A method, system, and computer program product are provided that represent complex point and area illumination (pure white or colored) in computer graphics shading and animation. An irradiance vector field table representative of an irradiance field for a scene to be rendered is generated and stored in a texture memory. During rendering, the scene is lit based on irradiance vectors in the irradiance vector field table. For each point being lit, a corresponding irradiance vector is generated from the irradiance vector field table. A vector operation is performed between the irradiance vector and a surface normal for the point to compute an irradiance energy component. The approaches with color illumination are: generating and storing a set of color irradiance vector field tables representative of an irradiance field at respective spectral wavelengths; storing an irradiance vector table and an irradiance color table; or storing an irradiance vector table and an irradiance color for the case of constant color.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Method, System, and Computer Program Product for Providing Illumination in Computer Graphics Shading and Animation

Background of the Invention

Field of the Invention

The present invention relates to computer graphics, and in particular, to shading and animating computer-generated images.

Related Art

Shading is an important component that contributes to the realism of a geometric scene to be rendered. The calculations associated with shading can be split into two parts: finding surface reflectance properties at each point being shaded, and computing the incident illumination at the point being shaded. Surface reflectance properties are used by computer graphics application programming interfaces (graphics APIs) to define how a surface dissipates light. For example, one graphics API, OpenGL™ by Silicon Graphics, Inc., supports five surface reflectance properties: diffuse and ambient properties, specular and shininess properties, and an emissive property. See, e.g., R. Fosner, *OpenGL™ Programming for Windows 95 and Windows NT* (Addison-Wesley Pub.: Reading, Mass. 1997), Chapter 9, "Colors, Materials, and Lights," pp. 181-201.

The shading calculation for an incident light component, however, is fairly primitive in current graphics hardware. As such the overall shading realism is diminished regardless of the sophistication in which surface reflectance properties are represented. In current commercial graphics shaders, incident light sources are only represented by point source lighting models. For example, OpenGL™ only supports directional lights, point light sources with varying distance-based falloff functions, and spot lights, which provide illumination in a cone.

One important type of point light source not yet represented in commercial graphics shaders is a point light source with emission varying by direction, as described by the Illuminating Engineering Society (IES) standard luminaire file format. The IES standard luminaire file format is a standard representation of emission from complex emitters, assuming an observer is sufficiently far away that the source can be treated as a point emitter. An IES recommended standard file format for electronic transfer for photometric data, IES LM-63-1990, is available from the Illuminating Engineering Society of North America.

In contrast to point lights, real-world light sources are area sources. Current methods for computing incident light from an area light source are impractical for computer graphics applications and systems. Current techniques are either too inefficient and expensive for hardware or real-time applications, or are only effective for area luminaires with limited shapes or emission distributions. For example, Ashdown has developed a method for mapping the emission distributions from complex area light sources onto a sphere (Ashdown, I., *J. Illuminating Eng. Soc.* 22(1):163-180 (Winter 1993)). Outgoing emission at a number of points on the sphere is computed for each point shaded—the computational requirements of this technique are excessive for current graphics hardware. Stock has developed a similar method, where the directional distribution of a light source is computed at various points over a sphere (Stock, R.D., *Fourier Formulation of Illumination Optics and Computer-Automated Reflector Design*, Ph.D. thesis, CMU (1995)). This distribution is projected into a Fourier space, and the coefficients are interpolated over the source of the sphere to construct intermediate distributions at arbitrary points. However, the techniques of Ashdown and Stock are still impractical for hardware or real-time applications.

Snyder has investigated closed form expressions for incident illumination for various types of area light sources (Snyder, J., "Area light sources for real-time graphics," *Microsoft Res. Tech. Rep.* MSR-TR-96-11 (March 1996)); this work is based on a paper written by Arvo in SIGGRAPH '95 (Arvo, J., "Applications of irradiance tensors to the simulation of non-Lambertian phenomena," in *ACM SIGGRAPH '95 Conf. Proc.*, Cook, R., ed., Addison Wesley (Aug. 1995), pp. 335-342). However, Snyder's techniques are still not sufficiently efficient for hardware implementation, and are limited to a small set of types of emitters.

Sun *et al.* have investigated techniques for tabularizing the calculation of the differential area to area form-factor (Sun, J., *et al., Computer Graphics Forum* 12(4):191-198 (1993)). This computation is equivalent to finding the incident energy from a diffusely emitting light source to a point. This technique reduces computation by tabularizing some aspects of it, but still has significant computational demands and does not easily generalize to more general emitter shapes or non-diffuse emission distributions.

Algorithms based on radiosity have been used to generate images of environments, taking into account both light from area light sources as well as interreflections of light from non-emissive surfaces. For a general introduction to radiosity, see: Francois Sillion and Claude Puech, *Radiosity and Global Illumination*, Morgan Kaufmann Publ: USA, 1994. Radiosity algorithms typically compute a simulation of light transport in a time-consuming pre-processing step, and the result can then be displayed interactively. With radiosity techniques, the lighting solution is computed at a

fixed set of points in the environment independent of the viewing parameters used for display. This one set of solution points can never be sufficient to provide an accurate solution for all possible viewing parameters.

Although complete radiosity solutions are impractical for interactive viewing of scenes with area light sources, some of the techniques used in radiosity solutions offer possibilities for computing light from area sources. These techniques are too computationally expensive and too inflexible for general applicability.

One of the techniques commonly used in radiosity algorithms to compute incident light at a point (the form-factor computation) is based on computing a contour integral around the edges of a polygonal light source (Daniel R. Baum and Holly E. Rushmeier and James M. Winget, "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors", SIGGRAPH '89 Proceedings, p. 325-334.). Although this is a potentially exact computation of the incident light, it is currently too expensive to evaluate in an interactive system, and cannot easily be extended to account for lights with non-diffuse emission distributions.

Wallace *et al.* have proposed a more efficient but less accurate method to compute form-factors, based on a simplifying assumption that the light is disc shaped (John R. Wallace and Kells A. Elmquist and Eric A. Haines, "A Ray Tracing Algorithm for Progressive Radiosity", SIGGRAPH '89 Proceedings). This is not an exact computation, unless the light is disc shaped and the point where incident light is being computed is parallel to the emitter and directly above or below it. Furthermore, the computation is still not efficient enough for real-time applications and cannot be easily extended to account for lights with varying emission distributions.

Ward has developed algorithms for global illumination and area light source computation based on computing scalar irradiance values at points on surfaces and interpolating them to compute irradiance at nearby points. See, Ward, Gregory J., *et al.*, "A Ray Tracing Solution for Diffuse Interreflection", in *Computer Graphics (SIGGRAPH '88 Proc.)*; Ward, Gregory J. and Heckbert, P., "Irradiance Gradients" in Third Eurographics Workshop on Rendering, May 1992; Ward,Gregory J., "The RADIANCE Lighting Simulation and Rendering System" in *SIGGRAPH '94 Proc.*, Orlando, Florida, July 24-29, 1994. Although these algorithms produce high-quality lighting simulations and naturally adapt to the rate at which irradiance is changing in a scene, they depend on storing irradiance estimates in a spatial data structure that is too complex for interactive applications, and the irradiance estimates are combined and weighted by a function that is also too complex for interactive applications. This algorithm doesn't store irradiance in a vector form, and

doesn't tabularize irradiance into a regular table structure which is necessary for efficient look up operations in hardware.

Other volumetric illumination representations have been developed by Chiu, K., *et al.*, "The Light Volume: An Aid To Rendering Complex Environments", Eurographics Workshop on Rendering, 1996 and Greger, G., *et al.*, "The Irradiance Volume", to appear in IEEE Computer Graphics and Applications, 1997.) These approaches also store representations of scene irradiance in a spatial data structure and interpolate them for intermediate points in the scene. These algorithms also require costly look-ups and significant computation to determine irradiance at points being shaded, making them unsuitable for interactive applications. This algorithm doesn't store irradiance in a vector form, and doesn't tabularized irradiance into a regular table structure which is necessary for efficient look up operations in hardware.

An efficient technique for computing incident light from complex point or area luminaires is needed. More complex luminaires are important for simulating more realistic scenes. A surface illuminated by a large area light source has a significantly softer appearance than if it were illuminated by a point source. Area light sources can also model common light sources (e.g., fluorescent tubes) more accurately than point sources.

Thus, a computer graphics method and system for shading and animation is needed which can efficiently compute incident light from point and/or area light sources at interactive rates. An efficient and general computing technique is needed which can compute incident light from a wide variety and range of point and area light sources in a rendered scene including sources with potentially complex emission distribution functions. Light sources also need to be represented when given data in the form of point light sources with emission varying by direction, as described by the IES standard luminaire file format.

Summary of the Invention

A method, system, and computer program product are provided that represent complex point and area illumination in computer graphics shading and animation. The complex point and area illumination can be pure white or colored.

An irradiance vector field table representative of an irradiance field for a scene to be rendered is generated and stored. During rendering, the scene is then lit based on irradiance vectors in the irradiance vector field table. For each point being lit, a corresponding irradiance vector is looked-up in the irradiance vector field table. A vector operation is performed between the irradiance vector and a surface normal for the point to compute an irradiance energy component. The irradiance energy component is then used in determining a final shade value for the point.

In one example hardware embodiment, a look-up unit looks up and interpolates irradiance vectors for selected points in the irradiance vector field table to compute an interpolated irradiance vector. A dot product unit calculates a dot product between the interpolated irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component. The irradiance energy component is used to determine the surface color of the point. For example, the irradiance energy component can be used in conjunction with the surface material properties to compute the surface color of the point.

According to a further feature of the present invention, a three-dimensional irradiance vector field table representative of a three-dimensional irradiance field can be generated and stored in any type of memory including, but not limited to, texture memory. The irradiance vector field table can represent an irradiance field for one or more point light sources including, but not limited to, a point light source defined by a luminaire file format. The irradiance vector field table can also represent an irradiance field for area light sources including, but not limited to, direct area emitters, such as, light bulbs, fixtures, and other area luminaires, and indirect area emitters, such as, secondary reflections from objects in a scene or a combination of both direct and indirect area emitters.

According to another embodiment of the present invention, multiple rendering passes are used to render a computer graphics image representing illumination from a complex point or area source. An irradiance vector field table is generated and stored in a texture memory. A first image is rendered using the irradiance vector field table to color pixels and stored in a first frame buffer. A second image is rendered and stored in a second frame buffer. The second image has surface normals encoded in red, green, and blue (RGB) color components.

To calculate a dot product, the first and second images are multiplied to obtain a third image. Components in the third image are accumulated to obtain a fourth image. The scene is rendered with any color and/or texture (not including the illumination represented by the irradiance field table) to obtain a fifth image. The fourth and fifth images are multiplied to obtain a final display image. This decoupling of the incident lighting component and the surface shading component is similar to the decoupling of incident lighting and texturing used to include texture mapping in a radiosity system (Gershbein, R., *et al.*, "Textures and Radiosity: Controlling Emission and Reflection with Texture Maps" in *SIGGRAPH '94 Proc.*).

With respect to color illumination with complex point or area sources, three approaches are discussed. In a first approach, a set of color irradiance vector field tables representative of an irradiance field at respective spectral wavelengths for a scene to be rendered are generated and stored in a texture memory. During rendering, the scene is lit based on a set of color irradiance vectors retrieved from the color irradiance vector field tables. For each point being lit, a corresponding set of irradiance vectors are generated from the color irradiance vector field tables. Vector operations are performed between each color irradiance vector and a surface normal for a shading point to compute an irradiance energy component. In one example hardware embodiment, for each color, a look-up unit looks up and interpolates color irradiance vectors close to a selected point in a respective color irradiance vector field table. A dot product unit calculates a dot product between each interpolated color irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component for use in shading. A set of multi-resolutional color irradiance field tables covering progressively larger regions of a scene at progressively coarser resolutions can be used.

A more efficient second approach stores an irradiance vector table and an irradiance color table. The irradiance vector field table is representative of an irradiance field for a scene to be rendered. The irradiance color table is representative of the color illumination at a spectral wavelength. During rendering, the scene is then lit based on irradiance vectors in the irradiance vector field table and irradiance colors in the irradiance color table.

For each point being lit, a corresponding irradiance vector is looked-up in the irradiance vector field table. This look-up operation can be a direct look-up or can involve interpolation of multiple irradiance vectors close to a point of interest. A vector operation is performed between the irradiance vector and a surface normal for the point to compute an irradiance energy component. An irradiance color is looked-up in the irradiance color table. This look-up operation can be a direct

look-up or can involve interpolation of multiple irradiance colors close to the point of interest. The irradiance energy component is then modulated using the irradiance color.

In one example hardware embodiment, a look-up unit looks up and interpolates irradiance vectors at points near the shading point in the irradiance vector field table to compute an interpolated irradiance vector for the shading point. A dot product unit calculates a dot product between the interpolated irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component. The look-up unit also looks up and interpolates irradiance colors at points near the shading point in the irradiance color table to compute an interpolated irradiance color. A surface modulator unit includes a multiplier that modulates the irradiance energy component by the interpolated irradiance color.

A third approach for the special case of constant color need only store an irradiance vector field table and an irradiance color. During rendering, the scene is then lit based on irradiance vectors in the irradiance vector field table and the stored irradiance color.

In particular, for each point being lit, a corresponding irradiance vector is looked-up in the irradiance vector field table as in the second approach. This look-up operation can be a direct look-up or can involve interpolation of multiple irradiance vectors close to a point of interest. A vector operation is performed between the irradiance vector and a surface normal for the point to compute an irradiance energy component. The irradiance energy component is then modulated using the stored irradiance color.

In one example hardware embodiment, a look-up unit looks up and interpolates irradiance vectors at points near the shading point in the irradiance vector field table to compute an interpolated irradiance vector for the shading point. A dot product unit calculates a dot product between the interpolated irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component. A surface modulator unit includes a multiplier that modulates the irradiance energy component by the stored irradiance color.

According to a further feature of the present invention, the irradiance vector field table can represent an irradiance field for one or more point light sources including, but not limited to, a point light source defined by a luminaire file format. The irradiance vector field table can also represent an irradiance field for area light sources including, but not limited to, direct area emitters, such as, light bulbs, fixtures, and other area luminaires, and indirect area emitters, such as, secondary reflections from objects in a scene or a combination of both direct and indirect area emitters. Likewise, the irradiance color vector field table can represent an irradiance field at a respective

spectral wavelength of a complex point or area illumination source. In one preferred embodiment, a set of three irradiance color vector field tables are used covering red, green, and blue wavelengths.

Because light and colored light are represented in a vector field, standard vector field algebraic operations can be performed on the field. For example, the fields for two light sources can be added together to find a single field that represents their combined illumination. This can make it possible to incorporate lighting from a number of light sources without increasing the computation that must be performed when shading calculations are performed.

According to a feature of the present invention, the irradiance vector field table (or the irradiance color vector field table(s)) are generated to account for shadows, or participating media, such as fog or smoke. In one embodiment, visibility information is encoded in the irradiance vector field table (or the irradiance color vector field table(s)) to represent shadows and/or participating media in the scene.

In another embodiment of the present invention, irradiance vector field tables (or irradiance color vector field tables) are computed without regard to shadows. Shadow maps are computed for a number of points on an emitter surface. The computed shadow maps are then used as in conventional shading calculations involving shadow maps as an approximation of how much of the light source is visible to the point being shaded. In particular, after a dot product operation is performed to obtain an irradiance energy component, the irradiance energy component is scaled using shadow map(s). Alternatively, standard back projection or shadow volume algorithms can be used to compute light source visibility from a point being shaded. The computed visibility is then used to scale the irradiance energy component.

According to a further feature of the present invention, a set of irradiance vector field tables (or a set of irradiance color vector field tables) are generated. Each table encompasses a larger volume of space. This reduces storage requirements and the number of computations needed to generate the table by taking advantage of situations where the irradiance field changes more slowly the farther away one is from the light source.

In one example implementation, a set of irradiance vector field tables (or a set of irradiance color vector field tables and/or irradiance color tables) are generated and stored as a clip-map (e.g., three-dimensional clip-map) to provide adaptive level-of-detail and to save texture memory. In the clip-map, the set of multi-resolutional irradiance field tables represent equal-size memory regions (also called tiles) which cover progressively larger regions of a scene at progressively coarser resolutions.

According to another embodiment of the present invention, multiple rendering passes are used to render a computer graphics image representing illumination with color from a complex point or area source. An irradiance vector field table is generated and stored in a texture memory. An irradiance color table is also generated and stored as a texture. A first image is rendered using the irradiance vector field table to color pixels and stored in a first frame buffer. A second image is rendered and stored in a second frame buffer. The second image has surface normals encoded in red, green, and blue (RGB) color components.

To calculate a dot product, the first and second images are multiplied to obtain a third image. Components in the third image are accumulated to obtain a fourth image. The scene is rendered with any color and/or texture (not including the illumination represented by the irradiance field table) to obtain a fifth image. The fourth and fifth images are multiplied to obtain a frame buffer display image.

The scene is also rendered using the stored irradiance color texture to obtain a sixth image. The sixth image and the frame buffer display image are multiplied to obtain the final frame buffer display image. This decoupling of the incident lighting component and the surface shading component is similar to the decoupling of incident lighting and texturing used to include texture mapping in a radiosity system (Gershbein, R., *et al.*, "Textures and Radiosity: Controlling Emission and Reflection with Texture Maps" in *SIGGRAPH '94 Proc.*).

Thus, the present invention provides an efficient and general technique for computing incident light (pure white or colored) from point or area luminaires at interactive rates. Complex point and area luminaires can be simulated efficiently to provide high-quality shading, animation, and more realistic scenes with illumination. Common area light sources, such as, incandescent bulbs,. fluorescent tubes, and lighting fixtures, and secondary reflections, can be represented in computer graphics shading and animation. Efficient hardware implementations are realized which compute incident illumination light for one or more area light sources yet require minimal additional hardware in a rendering pipeline.

Brief Description of the Figures

The accompanying drawings, which are incorporated herein and form part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to enable a person skilled in the pertinent art to make and use the invention.

FIG. 1 is a flowchart of a routine for providing illumination in computer graphics shading according to one embodiment of the present invention.

FIG. 2A is schematic diagram of an example irradiance vector field according to the present invention.

FIGs. 2B and 2B-1 are an irradiance field table representing the example irradiance vector field of FIG. 2A according to the present invention.

FIG. 3 is a flowchart of a multiple rendering passes routine for providing illumination according to the present invention.

FIG. 4A is a drawing showing multiple areas within a scene.

FIG. 4B is a drawing showing the multiple areas of FIG. 4A covered by a set of multi-resolutional irradiance field tables according to the present invention.

FIG. 5 is a flowchart of a routine for providing illumination using a set of multi-resolutional irradiance field tables in computer graphics shading according to another embodiment of the present invention.

FIG. 6 is a graphics processing pipeline for supporting illumination according to the present invention.

FIG. 7 is a block diagram illustrating a graphics subsystem for supporting illumination according to the present invention.

FIG. 8 is a block diagram illustrating additional hardware for supporting illumination according to the present invention.

FIG. 9 shows an example computer system including the graphics subsystem of FIG. 7.

FIGs. 10A and 10B are color images showing a scene illuminated by a circular source and a disc source, drawn in a simulation of the present invention.

FIG. 11 is a flowchart of a routine for providing illumination with color in computer graphics shading according to a first color embodiment of the present invention.

FIG. 12 is a flowchart of a routine for providing illumination with color in computer graphics shading according to second color embodiment of the present invention.

FIG. 13 is a flowchart of a multiple rendering passes routine for providing illumination according to one example implementation of the second color embodiment of the present invention.

FIG. 14 is a block diagram illustrating additional hardware for supporting illumination another example implementation of the second color embodiment of the present invention.

FIG. 15 is a flowchart of a routine for providing illumination of constant color according to a third color embodiment of the present invention.

The present invention is described with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference number identifies the drawing in which the reference number first appears.

Detailed Description of the Preferred Embodiments

TABLE OF CONTENTS

1. Overview

The present invention uses a tabularized vector irradiance field to compute incident color or pure white illumination light representative of a point or area light source in computer graphics shading and animation. The vector irradiance field is used in the present invention to encode the emission distributions from complex luminaires and compute incident light at points where shading calculations are performed. The vector irradiance field is known to illumination engineers in the electrical lighting field. See, Moon, P. and Spencer, D.E., *The Photic Field*, MIT Press, Cambridge, MA (1981)). The vector irradiance field is not as well known in computer graphics, though Arvo has based some closed-form illumination expressions on a vector irradiance field (Arvo, J., "The irradiance Jacobian for partially occluded polyhedral sources," in *Proc. ACM SIGGRAPH '94*, Glassner, A., ed., ACM Press (July 1994), pp. 343-350) and Hausner has investigated finding a spherical harmonic expansion of the field (Hausner, A., "Multipole Expansion of the Light Vector," *IEEE Transactions on Visualization and Comp. Graphics 3*(1) (Jan.-March 1997), pp. 12-22).

Prior to the present invention, however, there has been no teaching or suggestion to combine irradiance vector field art with commercial computer graphics shading art to efficiently represent light sources. Despite the commercial demand for an efficient graphics shader that supports complex point or area light sources, there has been no teaching or suggestion to efficiently and generally represent a point or area light source in a commercial graphics shader using a tabularized irradiance vector field, as in the present invention.

The irradiance field represents the emission from an emitter as a vector field in three-dimensional space. As recognized by the inventors, the irradiance vector field has two related characteristics that lead to highly efficient computation in computer graphics shading:

- The computation to find incident irradiance (emitted energy from the luminaire) at a point in space is a dot product: the dot product of the surface normal at the point with the irradiance vector at the point.
- This computation is independent of the complexity of the luminaire: given a representation of the irradiance field, the actual geometry and emission distribution of the emitter are no longer needed for the lighting calculation.

According to an embodiment of the present invention, a tabularized form of the irradiance field is generated and stored in a three-dimensional table in an efficient, inexpensive hardware implementation. According to another embodiment of the present invention, a set of multi-resolutional irradiance vector field tables is used which provide more precision close to the emitter

(where the field changes rapidly), and less precision far away (where the field changes very slowly). The set of multi-resolutional irradiance vector field tables provides an adaptive level-of-detail and saves memory costs. In one example, the set of multi-resolutional irradiance vector field tables is a clip-map. Note an adaptive representation of illumination energy has previously been used to accelerate radiosity algorithms by Hanrahan, P., *et al.*, "A Rapid Hierarchical Radiosity Algorithm", Computer Graphics (*SIGGRAPH '91 Proc.*).

Calculations can be performed in fixed-point or floating point arithmetic. In one implementation, calculations are performed using fixed-point arithmetic. This fixed-point implementation currently runs at 15fps (frames per second) on an Infinite Reality graphics machine sold by Silicon Graphics, Inc., a rate fast enough to support real-time computer graphics rendering and smooth animation.

2. *Terminology*

The term "area light source" is used to refer to any light source having an area of light emission in a computer graphics scene. An area light source can include a direct area emitters, such as, light bulbs, fixtures, and other area luminaires, and indirect area emitters, such as, secondary reflections from objects in a scene.

The terms "shading model", "lighting model", "illumination model", "global illumination model", and equivalents thereof, are used herein interchangeably to refer to any model or lighting equation that describes factors for determining a surface's color. The present invention can be applied with any illumination model including, but not limited to, the illumination models described herein as examples and those described in Foley *et al.*, *Computer Graphics: Principles and Practice, Second Ed. in C* (Addison-Wesley Publ.: Reading, MA), 1996 (incorporated in its entirety herein by reference), in particular see, chapter 16, pp. 721-814; and Beider *et al.*, *OpenGL Programming Guide*, Silicon Graphics, Inc. (Addison-Wesley Publishing Co., 1993), chapter 6, pp. 157-194 (incorporated herein by reference).

With respect to color illumination, the routines and systems described herein can be extended to include color sources. As used herein, the term "color" will generally refer to any non-pure white color or combinations of colors. The term "color source" includes sources having a constant color surface or a varying colored surface and sources where the irradiance is a function of wavelength, such as, sources which emit in a space that scatters different wavelengths of light differently.

3. *Example Environment*

The present invention is described in terms of an example computer graphics processing environment. As described herein, the present invention can be implemented as software, firmware, hardware, or any combination thereof. In one preferred implementation, the present invention provides an efficient hardware simulation of complex point and area light sources.

Given the description herein, it would be apparent to one skilled in the relevant art(s) to implement the present invention in any computer graphics application, application programming interface (API), or system including, but not limited to, a computer graphics processor (single chip or multiple chips), high-end to low-end graphics workstations, gaming platforms, systems and consoles, network architectures (e.g., client/server, local, intermediate or wide area networks), and virtual machine (e.g., a Java-created application). Any computer graphics architecture can be used including, but not limited to, an Open GL™ graphics API architecture (including but not limited to Infinite Reality, Indigo^2, Indy, Onyx, or O_2 graphics workstations manufactured by Silicon Graphics, Inc.) and raster graphics architectures such as those described in Foley *et al.*, chapter 18, pp. 855-922 (incorporated herein by reference above). Of course, these specific computer graphics systems are recited as examples which can support the present invention, but do not necessarily have to have the present invention to operate. For example, an Open GL™ graphics API architecture does not require use of the present invention. In other words, an Open GL™ graphics API architecture can provide illumination without representing light source(s) with an irradiance vector field, as described herein with respect to the present invention.

Description in these terms is provided for convenience only. It is not intended that the invention be limited to application in this example environment. In fact, after reading the following description, it will become apparent to a person skilled in the relevant art(s) how to implement the invention in alternative environments.

4. *The Irradiance Field-Theory*

The irradiance vector at a point P having components (P_x, P_y, P_z) generated by a light source is:

$$\vec{\Phi}(P) = \frac{1}{4\pi}\int_A \frac{1}{r^2} L_e(x_A, \omega)\vec{V}\cos\theta Vis(P,\ x\_A)Vol(P,\ x\_A)\ dA$$

where the integral is over the area A of the emitter, L_e is the emitted radiance at a point x_A on the emitter in the direction ω, r^2 is the square of the distance r from P to x_A, and V is the normalized vector from x_A to P. ω and θ, the angle between V and the emitter's surface normal at x_A are determined by V. Vis(P, x_A) is a visibility term, with a value of 1 if the points P and x_A are visible to each other and a value of 0 if there is an occluder between them. Vol(P, x_A) represents volume scattering effects of attenuating media between P and x_A; it can include both attenuation of light that is emitting from x_A before it reaches P as well as in-scattering of light from other points on the emitter before it reaches P.

The incident irradiance at the point P is computed by calculating the dot product of the surface normal at P and the irradiance field vector at P. As long as the emitter is not partially hidden by the horizon at P, this computation is exact. If the light source is partially hidden below the horizon at the point being shaded, the computation will report that more light is arriving than is actually arriving; however, this is not a major shortcoming because the amount of energy arriving when the light is close to the horizon is typically low.

The irradiance field for a point light source at origin (0,0,0) with uniform radiant intensity in all directions in a vacuum with no occluding objects present is trivially found:

$$\vec{\Phi}(P)=\frac{(P_x,P_y,P_z)}{r^3}$$

where

$$r=\sqrt{P_x^2+P_y^2+P_z^2}.$$

5. *Computing the Irradiance Field*

The field for an IES point source luminaire can be easily computed by taking the field for a point source and scaling the vectors based on the emission in the direction of a point in the field from the source. Computing the irradiance field for more complex emitting geometry and more complex emission distributions is more complicated than for point sources.

Finding the field for area light sources requires computing the integral above: closed form solutions exist for polygonal sources (Arvo, J., "The irradiance Jacobian for partially occluded polyhedral sources," in *Proc. ACM SIGGRAPH '94*, Glassner, A., ed., ACM Press (July 1994), pp. 343-350, incorporated herein in its entirety by reference), and disc sources (Moon, P. and Spencer,

D.E., *The Photic Field*, MIT Press, Cambridge, MA (1981), incorporated herein in its entirety by reference) that have uniform emission distributions. The irradiance field for more complex combinations of geometry and emission distribution functions will not always have a closed form solution. In this case, however, numeric integration methods should converge quickly; the irradiance field is typically smooth and slowly changing.

6. Shading Based on an Irradiance Vector Field

FIG. 1 shows a shading routine 100 for providing complex point or area illumination according to one embodiment of the present invention. First, an irradiance vector field table is generated (step 110). The irradiance vector field table has entries representing the vectors in an irradiance field. The irradiance vector field table is a three-dimensional table. Preferably, the irradiance vector field table is pre-computed and stored in a memory (such as a texture memory) prior to rendering. The present invention, however is not so limited, as the irradiance vector field table can be calculated and stored in a memory (such as a texture memory) during rendering depending upon processing power, time, and other design criteria. One-dimensional and/or two-dimensional tables can also be used.

Lighting stage 115 includes steps 120-140. Lighting stage 115 lights (or shades) one or more points in an object being rendered using irradiance field data stored in step 110. In general, for a point in emitter space P_e , to compute lighting and color, the irradiance vector at the point is computed by looking it up in the irradiance field table (step 120). For example, the eight vectors in the table that surround the point of interest P_e are found and trilinearly interpolated to find an irradiance vector I_e at the point P_e.

Next, an irradiance energy component E is computed based on a vector operation between irradiance vector I_e and a surface normal N_e at point P_e (step 130). In one preferred embodiment, the vector operation consists of calculating a dot product between irradiance vector I_e and a surface normal N_e at point P_e to find incoming irradiance energy component E. The irradiance energy component E is then used in any lighting equation or shading model to modulate surface color (step 140).

In one implementation where polygons (e.g. triangles) are rasterized, steps 120 and 130 are calculated on per-vertex basis (and interpolated inside the triangle), while step 140 is carried out on a per-pixel basis. However, the present invention is not intended to be so limited. In general, steps

120 to 140 can each be computed on a per-vertex and/or per-pixel basis, as would be apparent to a person skilled in the relevant art(s) given this description.

For image rendering algorithms (such as ray tracing or the REYES algorithm) that are not based on rasterizing polygons, shading calculations can be performed at varying rates and at varying locations. Application of the present invention to other rendering algorithms would be apparent to a person skilled in the relevant art(s) given this description.

FIG. 2A is schematic diagram of a two-dimensional slice of an example irradiance vector field 210 for a disc source 200, according to the present invention. FIGs. 2B and 2B-1 show an irradiance field table 220 representing the example irradiance vector field 210 of FIG. 2A. For clarity, irradiance vector field 210 only includes 64 points. At each point, is a vector representing an irradiance field vector. Accordingly, irradiance field table 220 is a two-column table having sixty-four entries (or rows). Each entry includes position and irradiance field vector coordinates of a respective point and vector. The position is given by three spatial coordinates in a coordinate space, such as, the coordinate space of the complex point or area light source (also called emitter coordinate space). The vector coordinates represent three components defining the irradiance field at the point.

In the above description of FIG. 1, the irradiance vector field table was generated in an emitter space. In other words, when the table of irradiance vectors for a light source is computed, the emitter is at the origin with some orientation. The point P_e being shaded and surface normal N_e were also in emitter space (or are transformed to emitter space from world space or any other coordinate space prior to lighting).

To position the emitter differently when rendering, it can be transformed with a standard 4x4 geometry matrix. Two adjustments are then made in shading routine 100 to ensure calculations are performed in a common coordinate space. First, each point being shaded is transformed by the same coordinate transformation as the emitter, before the point coordinates are used to index the irradiance vector field texture. Second, the same transformation is applied to the surface normal at the point being shaded before the dot product is computed.

a. Pseudocode

To summarize the algorithm of the present invention, this section has two short pseudocode representations of its execution. In one embodiment described below with respect to FIG. 3, either code can be implemented using frame-buffer operations, so that it is effectively running in parallel at all points being shaded.

b. Example 1

```
float dot (Vector V Normal N)
{
return (V.x * N.x) + (V.y * N.y)+ (V.z * N.z);
}
/*      transformation that is applied to the emitter */
Transformation world2emitter;
/*      multi-resolution representation of the
irradiance field */
VectorField irradiance-field;

Color shade_pixel (Point P, Normal N, Color C)
{
        Vector irradiance_vector;
        float modulate;

        /*      transform the point being shaded to the
                coordinate space of the emitter*/
        P =     world2emitter.transform(P);
        N =     world2emitter.transform(N);

        /*      use this point to look up the irradiance
                vector*/
        irradiance_vector = irradiance_field.lookup(P);

        /*      compute the dot product to find the incoming
                irradiance */
        modulate = dot (irradiance_vector, N);

        /*      use the result to modulate the surface's color */ return C * modulate;
}
```

c. Example 2

```
/*      transformation that is applied to the emitter
        and its inverse */
Transformation world2emitter;
Transformation emitter2world;

/*      multi-resolution representation of the
        irradiance field */
VectorField irradiance-field;

Color shade_pixel (Point P, Normal N, Color C)
{
        Vector irradiance_vector;
        double modulate;
```

```
        /*      transform the point being shaded to the
                coordinate space of the emitter*/
        P =     world2emitter.transform(P);

        /*      use this point to look up the irradiance
                vector*/
        irradiance_vector = irradiance_field.lookup(P);

        /*      transform the irradiance vector found back into world space (which is the space
                that the surface normal is in. */
        irradiance_vector =
                emitter2.world.transform (irradiance_vector);
        /*      compute the dot product to find the incoming
                irradiance */
        modulate = dot (irradiance_vector, N);

        /*      use the result to modulate the surface's color */ return C * modulate;
}
```

7. Shadow and Media Effects

Computing shadows and/or participating media effects from light sources is also important for realism. In one example, visibility information representing shadows and/or participating media, such as, fog or smoke, is further encoded in the tabularized field. For example, in regions of space that were in shadow, the vectors would be zero. Under this approach, shading quality for shadows would be less coarse in higher resolution irradiance vector field tables.

The effects of participating media (such as fog or smoke) on light as it travels through space can be incorporated into the vector table as well. The irradiance vector field table is generated as described above without regard to participating media. Each vector is then scaled by a term that represents how much light is to be attenuated and how much light has been scattered at each point in the table. For example, if the scene and the light source are surrounded by homogeneous fog that absorbs 10% of the light along each unit distance and if only attenuation due to the fog is considered, the volume term in the irradiance field equation would be Vol(P, x_A) = 0.9 ^ (distance(P, x_A)). Mobley has developed closed form expressions for volumetric scattering that include multiple scattering of light as well as attenuation that could also be used for the volume term. See, Mobley, "Light and Water: Radiative Transfer in Natural Waters" Academic Press, 1994, incorporated herein in its entirety by reference. This makes it possible to efficiently simulate the effects of complex scattering of light by media when the table is generated. There is no additional cost to including media effects when a point is being shaded.

In another embodiment, shadow maps can be used with the present invention. Shadow maps are effective for computing shadows from point sources. In particular, shadow maps can be computed for a number of points on the source. Irradiance vector field table(s) can be computed with or without regard to shadows or participating media effects. After the irradiance energy component is found in a dot product operation, the irradiance energy component is scaled with an estimate of what fraction of the emitter is visible based on information from the shadow maps. See, Williams, L., "Casting curved shadows on curved surfaces," in *Computer Graphics (SIGGRAPH '78 Proc.)*, vol. 12 (August 1978), pp. 270-274; and Reeves, W.T., *et al.*, "Rendering antialiased shadows with depth maps," in *Computer Graphics (SIGGRAPH '87 Proc.*, vol. 21, Stone, M.C., ed., (July 1987), pp. 283-291, both of which are incorporated by reference herein. Note that shadow maps (and particularly multiple shadow maps used for area light sources) are only an approximation to the visibility between the emitter and the receiving point.

Alternatively, back-projection algorithms or shadow volume algorithms can be used instead of shadow maps to compute light source visibility and then to scale the irradiance energy component. However, back projection algorithms are typically more inefficient than computing light source visibility with shadow maps. See, e.g., the back-projection algorithms described by Bouknight, W. J, and Kelley, K. C, "An Algorithm for Producing Half-Tone Computer Graphics Presentations with Shadows and Movable Light Sources," Proceedings of the Spring Joint Computer Conference, AFIPS Press, Montvale, NJ, 1970, pp. 1-10 (incorporated in its entirety herein by reference); and Atherton, P. R., Weiler, K., and Greenberg, D, "Polygon Shadow Generation," Proceedings of SIGGRAPH 78, pp. 275-281 (incorporated in its entirety herein by reference); and the shadow volume algorithm described by Crow, F. C, "Shadow Algorithms for Computer Graphics," Proceedings of SIGGRAPH 1977, pp. 242-247 (incorporated in its entirety herein by reference). Note that shadow maps (and particularly multiple shadow maps used for area light sources) are only an approximation to the visibility between the emitter and the receiving point.

8. Fixed Point and Floating Point Implementations

The present invention can be implemented on computer systems that use fixed-point and/or floating point arithmetic. In particular, routine 100 is performed using fixed point arithmetic when a graphics system, such as the current InfiniteReality system by Silicon Graphics, Inc., does not support floating point values in all stages of the hardware rendering pipeline. As a result, a number

of additional computations are necessary, primarily to compute the dot product, which is more complicated in fixed point arithmetic.

High-quality shading images of a variety of scenes with a variety of light sources using a simulator having a floating point frame buffer have been obtained. For example, FIGs. 10A and 10B are color images showing a scene illuminated by a circular source and a disc source, drawn in a simulation of the present invention.

Routine 100 can be implemented in software, firmware, hardware, and in any combination thereof, as would be apparent to person skilled in the relevant art(s) given this description. For example, routine 100 can be implemented in a lighting and coloring stage of a graphics engine. In one example, per-vertex operations can be implemented primarily in software and per-pixel operations can be implemented primarily in hardware.

Examples of a graphics processing environment and architecture providing complex point and area illumination according to the present invention are described in further detail with respect to FIGs. 6 to 9.

9. Multiple Rendering Passes

According to a further embodiment, the shading calculation described with respect to routine 100 is implemented by performing multiple rendering passes and frame buffer stores. FIG. 3 is a flowchart of a multiple rendering passes routine 300 for providing complex point and area illumination according to the present invention. In routine 300, a scene with a complex point or area light source is rendered three times to do the shading calculation described in FIG. 1. In step 310, an irradiance vector field table is stored in a 2D or 3D texture map as described earlier with respect to step 110.

In step 320, a scene is rendered using the stored 3D texture map to color pixels and the resultant first image is stored in a first framebuffer (FB1). In this framebuffer, the red, green, and blue components of each pixel are equal to the respective x, y, and z components of the irradiance vector at that point. For example, in step 320, a texture generator function (such as "texgen" in OpenGL™) can be used to map points in space to the 3D texture map that holds the irradiance field—this gives an image with a trilinearly interpolated irradiance vector for each visible point that is being shaded. Next, in step 330, the scene is rendered with surface normals encoded in red, green, and blue (RGB) color components and the resultant second image is stored in a second framebuffer (FB2).

The two framebuffer images (FB1, FB2) are used to compute the dot product $I_e \bullet N_e$ (steps 340 and 350). In step 340, the contents of framebuffers, FB1 and FB2, are multiplied to obtain a third image FB3. In step 350, components of pixels in the third image are added together with the result stored in each of the red, green, and blue components to obtain a fourth framebuffer image (FB4). Finally, the scene geometry in framebuffer image FB4 is rendered with any further colors and texture-maps to obtain a fifth framebuffer image FB5 (step 360). The fourth and fifth framebuffer images FB4 and FB5 are multiplied to obtain a final framebuffer display image (step 370). In this way, the fifth image is modulated with the per-pixel results of the dot product represented in the fourth image.

As would be apparent to a person skilled in the relevant art(s) given this description, this six-pass algorithm can be modified to use more or fewer passes, depending on the resources available in a given hardware architecture, or depending on the speed of the operations in the passes. For example, an architecture might support a dot product operation between two framebuffers in a single pass, which would allow steps 340 and 350 to be combined into a single pass.

Multiple rendering routine 300 can be implemented as software, firmware, hardware, or any combination thereof in a computer graphics processor. In one preferred implementation, routine 300 is written as an OpenGL™ program for execution on any OpenGL™ machine including, but not limited to, graphics workstations manufactured by Silicon Graphics, Inc.

10. Clip Maps

According to another embodiment of the present invention, clip-maps are used to store a 2D or 3-D irradiance vector field table. Clip-maps can provide an adaptive level-of-detail and great savings in the amount of texture memory.

The inventors further recognize that the smoothness of irradiance field can be taken advantage of by using clip-maps for a 2-D or 3-D irradiance vector field table. Irradiance vectors change quickly close to a light source. However, at locations further from the light source, irradiance vectors eventually approach a radially symmetric field - as a point source would generate. (Illumination engineers have a rule of thumb that says that an area light source can be treated as a point source for points a distance of five times the light source diameter away (Ashdown, I., *J. Illuminating Eng. Soc.* 22(1):163-180 (Winter 1993)).

Clip-mapping cleanly deals with points that are far away from the emitter, without needing a special case for points outside of a finite-sized table. In clip-mapping, a set of two- or three-dimensional irradiance vector field tables are generated to hold the field for a scene. Each successive irradiance vector field table spans a larger volume of space at a lower or coarser resolution. The coarsest level, corresponding to irradiance received very far away from the emitter encompasses all of space for a scene and has zero-length vectors. Clip mapping also allows these multiple levels of detail to be cleanly specified and organized.

For example, as shown in FIG. 4A, a scene having an area light source 401 can be sub-divided into three different field regions 400, 422, 424 covering a progressively larger area of the scene. According to the present invention, irradiance vector field tables 410, 412, 414 are generated independently for the corresponding field regions 400, 422, 424, respectively. Each irradiance vector field table 410, 412, 414 spans a larger volume of space at a lower or coarser resolution. In this way, each irradiance vector field table 410, 412, 414 can be equal in size (e.g. 16x16 texels), resulting in a great memory savings compared to generating a complete set of multi-resolutional irradiance field vector tables where each table must cover the entire scene. Note that one-eighth of each table after the first

one (which covers the smallest volume of the scene at the finest resolution) doesn't need to be computed and stored, since its entries will never be accessed. Of course, the present invention is not limited to clip-maps only. A single table at a fixed resolution for an entire scene or a complete set of multi-resolutional irradiance field vector tables can be used.

FIG. 5 is a flowchart of a routine 500 for providing complex point and area illumination using a clip-map of multi-resolutional irradiance field tables in computer graphics shading according to another embodiment of the present invention. First, a set of multi-resolutional irradiance field tables is generated (step 510). The set of multi-resolutional irradiance field tables corresponds to a set of memory regions or tiles in a clip-map as described above with respect to FIGs. 4A and 4B. Preferably, the set of irradiance field tables is pre-computed and stored as a three-dimensional clip-map in a memory (such as a texture memory) prior to rendering. However, the present invention is not so limited as the irradiance field tables can be calculated and stored in a memory (such as a texture memory) during rendering depending upon processing power, time, and other design criteria.

Lighting stage 515 includes steps 522-540. Lighting stage 515 lights (or shades) one or more points in an object being rendered using irradiance field data stored in step 510. First, an irradiance table is selected which has a resolution (or level-of-detail) that covers a point in emitter space P_e.

Preferably, the irradiance table within the clip-map is selected which has the finest resolution that covers the point P_e (step 522).

Steps 524, 530, and 540 are similar to steps 120-140 described above with respect to FIG. 1. To compute lighting at its surface normal, and to compute its color, the irradiance vector at the point is computed by looking it up in the selected irradiance field table (step 522). For example, the eight vectors in the table that surround the point of interest are found and can be trilinearly interpolated to find the irradiance vector I_e at the point P_e. Other filtering, such as tricubic, can also be used to compute the irradiance vector I_e.

Next, an irradiance energy component E is computed based on a vector operation between irradiance vector I_e and a surface normal N_e at point P_e (step 530). For example, the vector operation can include calculating a dot product between irradiance vector I_e and a surface normal N_e at point P_e to find incoming irradiance energy component E. The irradiance energy component E is then used in any lighting equation or shading model to modulate surface color (step 540).

In one implementation, steps 522, 524, and 530 are calculated on per-vertex basis, while step 540 is carried out on a per-pixel basis. However, the present invention is not intended to be so limited. In general, steps 522 to 540 can each be computed on a per-vertex and/or per-pixel basis, as would be apparent to a person skilled in the relevant art(s) given this description.

The present invention can also adapt detail in the clip maps depending upon how quickly the field is changing. In the above description, clip map tiles are arranged concentrically around the emitter which is most effective assuming the irradiance field becomes smoother in all directions at the same rate. For a light source with highly directional emission, the clip map should reflect that the field is changing more quickly in some directions than in others (for example, the shape of the tables could become rectangular).

11. Animation

The present invention can also be used to provide illumination in computer graphics animation, as would be apparent to a person skilled in the relevant art(s) given this description. The emitter's transformation can be modified from frame to frame. This means that the emitter can be moved around the scene very efficiently without needing to recompute the irradiance vector field table. If the shape or emission properties of the emitter vary, however, some or all of the irradiance vector field table is updated. If the geometry in the scene is being animated, shading calculations are re-done for each frame as is typically required in rendering animated sequences. Thus, no

additional work is needed compared to what normally must be done when rendering animated sequences. See, e.g., Watt and Watt, *Advanced Animation and Rendering Techniques: Theory and Practice* (ACM Press, Addison-Wesley Publ.: New York, New York), 1992 (incorporated in its entirety herein by reference); and Foley *et al.*, chapter 21, pp. 1057-1081, incorporated herein by reference above.

12. Example Implementations

Examples of a graphics processing environment and graphics subsystem architecture providing complex point and area source illumination according to the present invention are described below with respect to FIGs. 6 to 9. These examples are illustrative and not intended to limit the present invention.

FIG. 6 is a block diagram of an example graphics processing pipeline environment 6000 for implementing routines 100, 300, and 500 as described above. Graphics processing environment 6000 can include but is not limited to an OpenGL™ operation as described in Beider *et al.*, *OpenGL Programming Guide, The Official Guide to Learning OpenGL*, Release 1, Silicon Graphics Inc. (Addison-Wesley Publishing Co., USA, 1993) and *OpenGL Reference Manual, The Official Reference Document for OpenGL*, Release 1, Silicon Graphics Inc. (Addison-Wesley Publishing Co., USA, 1992) (both of which are incorporated in their entirety herein by reference).

As shown in FIG. 6, graphics processing pipeline 6000 consists of a display list 6007, evaluator 6010, per-vertex operations and primitive assembly stage 6020, rasterization stage 6030, pixel operations stage 6040, texture memory 6050, per-fragment operations stage 6060, framebuffer 6070, and a computer graphics display unit 6080. Commands 6005 are input to the graphics processing pipeline 6000. Commands 6005 specify geometric objects to be drawn and control how the objects are handled during the various processing stages. Commands 6005 can be processed immediately through the pipeline 6000 or can be accumulated in display list 6007 for processing at a later time.

Evaluator 6010 approximates curve and surface geometry by evaluating polynomial commands of input values. During the next stage, per-vertex operations and primitive assembly stage 6020 processes geometric primitives. Geometric primitives are points, line segments, triangles, and polygons, all of which are described by vertices. Vertices are transformed and lit, and primitives are clipped to a viewport in preparation for the rasterization stage 6030.

Rasterization stage 6030 produces a series of framebuffer addresses and associated values using a two-dimensional description of a point, line segment, triangle, or polygon. Each fragment produced in rasterization stage 6030 is fed into the last stage, per-fragment operations stage 6060. Per-fragment operations stage 6060 performs the final operations on graphics data before the data stored as pixels in framebuffer 6070. These final operations can include conditional updates to the framebuffer 6070 based on incoming and previously stored Z values for Z buffering, blending of incoming pixel colors with stored colors, masking, and other logical operations on pixel values.

Input graphics data can be in the form of pixels rather than vertices. For example, an image used in texture mapping is processed in a pixel operation stage 6040. Pixel operation stage 6040 processes graphics data as pixels and stores a resulting texture map in texture memory 6050. Rasterization stage 6030 can then use the texture map stored in texture memory 6050 for performing texture processing. The output from pixel operations stage 6040 can also be applied directly to rasterization stage 6030 and merged with resulting fragments into framebuffer 6070 just as if the output was generated from geometric data.

In one embodiment of the present invention, the routine for providing complex point and area illumination 100 is implemented in graphics processing pipeline 6000 as follows. Texture memory 6050 is used to store the irradiance vector field table generated in step 110. Step 120 is processed as per-vertex operations in the per-vertex operations and primitive assembly stage 6020.

Steps 130 and 140 are processed on a per-pixel basis in pixel operations stage 6040. Pixel operations stage 6040 (or alternatively per-vertex operations and primitive assembly stage 6020) can output a color value based on diffuse and specular components calculated in step 140 to rasterization stage 6030 for processing.

FIG. 7 is a schematic diagram showing an example graphics subsystem 700 implementing complex area or point illumination in shading according to the present invention. Lighting and coloring module 710 includes an illumination module 720. A memory 730 stores a 2D or 3D irradiance vector field table (or set of multi-resolutional irradiance vector field tables) generated according to steps 110, 310, or 510 above. In one example, illumination module 720 implements routine 100 primarily in software as a program module that can be executed by a graphics processor or another processor. Various filtering functions can also used to interpolate between entries in the irradiance vector field table(s).

13. Efficient Hardware Simulation of Complex Point and Area Light Sources

FIG. 8 is a block diagram of illumination module 720 that implements routine 100 inexpensively in hardware according to another embodiment of the present invention. Illumination module 720 includes look-up unit 810, dot product unit 820, and a surface color modulation unit 830. A transformation unit (not shown) can be used to transform a point and surface normal from world space to emitter space. In general, each transformation of the point or surface normal requires 16 multiply operations and 12 add operations.

Look-up unit 810 computes an irradiance vector for a point using the stored irradiance vector table as described with respect to step 110. Look-up unit 810 then finds 8 pre-computed vectors adjacent to the transformed point in emitter space and performs a trilinear interpolation to obtain a lighting vector representative of the irradiance vector field. This look-up and interpolation requires 8 vector look-ups, 21 multiply operations, and 42 additions.

Dot product unit 820 computes the dot product of the vector output by look-up unit 810 and the surface normal in emitter space N. This dot product calculation only requires 3 multiply operations and 2 add operations. Dot product unit 820 then outputs the irradiance energy component E.

Surface color modulator unit 830 uses irradiance energy component E to modulate surface color in an operation that only requires 3 multiply operations. Surface color modulator unit 830 outputs an shading color representative of incident point or area illumination which is then output as the final shaded color or combined with further lighting and coloring operations in a shader.

An optional unit (not shown) for computing light visibility information (or occlusion), as described above, can used to scale the irradiance energy component E output from dot product unit 820 and/or to scale the shading color from result from surface color modulator unit 830.

Thus, in this embodiment, the present invention can represent complex point and area light sources inexpensively with little additional hardware.

14. Example GUI Computer System

FIG. 9 is a block diagram illustrating an example environment in which the present invention can operate. The environment is a computer system 900 that includes one or more processors, such as processor 904. Computer system 900 can include any type of computer graphics computer, virtual machine, processor (single bus, multiple bus, or bus-less processor(s)), workstation, and

network architecture. In one preferred implementation, an OpenGL machine can be used including, but not limited to, graphics workstations manufactured by Silicon Graphics, Inc.

Processor 904 is connected to a communications bus 902. Various software embodiments are described in terms of this example computer system. This description is illustrative and not intended to limit the present invention. After reading this description, it will be apparent to a person skilled in the relevant art(s) how to implement the invention using other computer systems and/or computer architectures.

Computer system 900 includes a graphics subsystem 903. Graphics subsystem 903 can be implemented as one or more processor chips. Graphics subsystem 903 can be included as part of processor 904 as shown in FIG. 9 or as a separate graphics engine or processor. Graphics data is output from the graphics subsystem 903 to the bus 902. Display interface 905 forwards graphics data from the bus 902 for display on the display unit 906.

Computer system 900 also includes a main memory 908, preferably random access memory (RAM), and can also include a secondary memory 910. The secondary memory 910 can include, for example, a hard disk drive 912 and/or a removable storage drive 914, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive 914 reads from and/or writes to a removable storage unit 918 in a well known manner. Removable storage unit 918 represents a floppy disk, magnetic tape, optical disk, etc., which is read by and written to by removable storage drive 914. As will be appreciated, the removable storage unit 918 includes a computer usable storage medium having stored therein computer software and/or data.

In alternative embodiments, secondary memory 910 may include other similar means for allowing computer programs or other instructions to be loaded into computer system 900. Such means can include, for example, a removable storage unit 922 and an interface 920. Examples can include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 922 and interfaces 920 which allow software and data to be transferred from the removable storage unit 922 to computer system 900.

Computer system 900 can also include a communications interface 924. Communications interface 924 allows software and data to be transferred between computer system 900 and external devices via communications path 926. Examples of communications interface 924 can include a modem, a network interface (such as Ethernet card), a communications port, etc. Software and data transferred via communications interface 924 are in the form of signals which can be electronic, electromagnetic, optical or other signals capable of being received by communications interface 924,

via communications path 926. Note that communications interface 924 provides a means by which computer system 900 can interface to a network such as the Internet.

Graphical user interface module 930 transfers user inputs from peripheral devices 932 to bus 902. These peripheral devices 932 can be a mouse, keyboard, touch screen, microphone, joystick, stylus, light pen, voice recognition unit, or any other type of peripheral unit.

The present invention is described in terms of this example environment. Description in these terms is provided for convenience only. It is not intended that the invention be limited to application in this example environment. In fact, after reading the following description, it will become apparent to a person skilled in the relevant art(s) how to implement the invention in alternative environments.

The present invention is preferably implemented using software running (that is, executing) in an environment similar to that described above with respect to FIG. 9. In this document, the term "computer program product" is used to generally refer to removable storage unit 918 or a hard disk installed in hard disk drive 912. These computer program products are means for providing software to computer system 900.

Computer programs (also called computer control logic) are stored in main memory and/or secondary memory 910. computer programs can also be received via communications interface 924. Such computer programs, when executed, enable the computer system 900 to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor 904 to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system 900.

In an embodiment where the invention is implemented using software, the software may be stored in a computer program product and loaded into computer system 900 using removable storage drive 914, hard drive 912, or communications interface 924. Alternatively, the computer program product may be downloaded to computer system 900 over communications path 926. The control logic (software), when executed by the processor 904, causes the processor 904 to perform the functions of the invention as described herein.

In another embodiment, the invention is implemented primarily in firmware and/or hardware using, for example, hardware components such as application specific integrated circuits (ASICs). Implementation of a hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant art(s).

15. Color Illumination

The complex point and area illumination light described above can be pure white. The irradiance vector described above has no spectral characteristics. In reality, a light source generates a distribution of irradiance vectors varying by the wavelength of the light they carry. In other words, to provide color illumination, the irradiance vector described above can be modified to be a function of wavelength. For example, the irradiance color vector at a point P at wavelength λ having components (P_x, P_y, P_z) generated by a light source is:

$$\vec{\Phi}(P,\lambda) = \frac{1}{4\pi}\int_A \frac{1}{r^2} L_e(x_A,\omega,\lambda)\vec{V}\cos\theta Vis(P,\ x_A)Vol(P,\ x_A)\ dA$$

where the integral is over the area A of the emitter, L_e is the emitted radiance at wavelength λ at a point x_A on the emitter in the direction ω, r^2 is the square of the distance r from P to x_A, and V is the normalized vector from x_A to P. ω and θ, the angle between V and the emitter's surface normal at x_A are determined by V. Vis(P, x_A) is a visibility term, with a value of 1 if the points P and x_A are visible to each other and a value of 0 if there is an occluder between them. Vol(P, x_A) represents volume scattering effects of attenuating media between P and x_A; it can include both attenuation of light that is emitting from x_A before it reaches P as well as in-scattering of light from other points on the emitter before it reaches P. The variable λ represents the wavelength of the emitter. Note, for clarity, the term "irradiance color vector" will be used hereafter to refer to an irradiance vector which is a function of wavelength.

Further, as in other graphics applications, the standard red, green, blue (RGB) approximation can be applied. In this case rather than having a single irradiance vector, combinations of red, green, and blue irradiance color vectors can be used.

In some applications it is desirable to have a color light source. For example, such colored light sources can be any light with a constant color that is not white, lights with varying color over their surface (e.g., a television screen), or lights through a non-homogeneous fog that scatters different wavelengths of light differently.

Three approaches to color illumination according to the present invention are provided. The first approach to color illumination is described primarily with respect to FIG. 11. The second approach to color illumination is described primarily with respect to FIGs. 12 to 14. These two approaches are similar in functionality and generally applicable to all color sources including constant color sources; however, the second approach requires less computations. A third approach

is directed to the special case of color sources having a constant color. The third approach can be implemented much more efficiently and with less computations than the first and second more general approaches. One example of the third approach is described with respect to FIG. 15.

a. First Approach for Color Illumination

FIG. 11 is a flowchart of a routine 1100 for providing illumination with color in computer graphics shading according to a first color embodiment of the present invention. Routine 1100 is similar to routine 100 described above, accordingly, only the modifications to support color illumination need be described in detail.

In step 1110, three color irradiance vector field tables are computed instead of one irradiance vector field table as described in step 110. For example, the three color irradiance vector field table can correspond to red, green, and blue color irradiance vector fields. In lighting stage 1115, at each shading point P_e, three color irradiance vectors (I_R, I_G, I_B) are computed through look-up operations to the three color irradiance field tables (step 1120). For each color, a look-up operation can be a direct look-up of a color irradiance vector or can involve an interpolation of color irradiance vectors close to a point of interest. Next, an irradiance energy component E is computed at each shading point P_e (step 1130). In particular, a vector operation (e.g., a dot product) is performed between each of the three color irradiance vectors (I_R, I_G, I_B) and a surface normal in emitter space. These three dot products can be summed to obtain the irradiance energy component E.

Finally, the irradiance energy component E computed in step 1130 is used in step 1140 to modulate the surface color of the shading point, as described previously with respect to step 140.

b. Second Approach for Color Illumination

The second approach optimizes the vector operation in step 1130 for the case of a dot product. In particular, the second approach recognizes that a dot product can be further optimized by taking advantage of the following property of the dot product of vectors: if r is a real number and v1 and v2 are vectors, then

((r*v1) dot (v2)) == r*(v1 dot (v2)).

Applying this dot product property to the three dot products of each of the three color irradiance vectors (I_R, I_G, I_B) and a surface normal in emitter space allows one irradiance vector table and one color table to be used. At each shading point in emitter space, then only one dot product is performed between an irradiance vector (with no spectral component) and a surface normal as described above with respect to step 130. The resultant irradiance energy component is then modulated by an interpolated color from the color table.

FIG. 12 is a flowchart of a routine 1200 for providing complex point or area illumination with color in computer graphics shading according to a second color embodiment of the present invention. In step 1210, one irradiance vector table and one color table is generated. The irradiance vector table is pre-computed as described above with respect to step 110. One color table is generated having one or more colors assigned to each point in emitter space. In one preferred embodiment, red, green, and blue components are assigned to each point in emitter space.

At lighting stage 1215, in step 1220, an irradiance vector I_e (with no spectral component) is computed as described above with respect to step 120. For example, the irradiance vector I_e can be easily retrieved from the irradiance vector field table generated in step 1210. A look-up operation or unit can be used to look-up the irradiance vector I_e corresponding to the point being shaded in emitter space. A direct look-up or interpolation of nearby irradiance vectors can be performed.

Likewise, in step 1222, an irradiance color is computed. For example, the irradiance color can be retrieved from the irradiance color table generated in step 1210. A look-up operation or unit can be used to look-up the irradiance color components (i.e. red, green, and blue) corresponding to the point being shaded in emitter space. An interpolator can also interpolate retrieved irradiance color components of nearby points to obtain an interpolated irradiance color value corresponding to the point being shaded in emitter space.

In step 1230, at each shading point in emitter space, then a dot product is calculation is performed between the irradiance vector I_e computed in step 1220 and a surface normal N_e as described above with respect to step 130. In step 1232, the resultant irradiance energy component E is modulated by the irradiance color computed in step 1222.

Finally, in step 1240, a modulated irradiance energy component E_c from step 1232 is used in any lighting equation or shading model to modulate surface color (similar to step 140 described above).

FIG. 13 is a flowchart of a multiple rendering passes routine 1300 for providing illumination according to one example implementation of the second color embodiment of the present invention. Routine 1300 is similar to routine 300 described above. Indeed, steps 1310, 3120, 1330, 1340, 1350,

1360, and 1370 are identical to steps 310, 320, 330, 340, 350, 360, and 370 described above and as would be apparent to a person skilled in the relevant art(s) do not need to be described further.

Steps 1312, 1362, and 1372 are added to provide color illumination. In step 1312, an irradiance color table is generated and stored as a 3D texture map (called the irradiance color texture). In step 1362, the scene is rendered with using the irradiance color texture stored in step 1312. This produces a sixth frame buffer image FB6 which is stored in a frame buffer. In step 1372, the sixth frame buffer image FB6 is then multiplied with the frame buffer display image output from step 1370 to obtain a final frame buffer display image.

FIG. 14 is a block diagram illustrating an illumination module in another example implementation of the second color embodiment of the present invention. The illumination module includes look-up unit 1410, dot product unit 1420, and a surface color modulation unit 1430. A transformation unit (not shown) can be used to transform a point and surface normal from world space to emitter space. In general, each transformation of the point or surface normal requires 16 multiply operations and 12 add operations.

Look-up unit 1410 computes an irradiance vector for a point using the stored irradiance vector table in a memory 730 as described with respect to step 110. Look-up unit 1410 finds 8 pre-computed vectors adjacent to the transformed point in emitter space and performs a trilinear interpolation to obtain a lighting vector I_e representative of the irradiance vector field. This look-up and interpolation requires 8 vector look-ups, 21 multiply operations, and 42 additions.

To provide color illumination, look-up unit 1410 retrieves corresponding irradiance color components (i.e. red, green, and blue) from memory 1412 corresponding to the point being shaded in emitter space. An interpolator in look-up unit 1410 interpolates irradiance color components from one or more nearby points to obtain an irradiance color value corresponding to the point being shaded in emitter space. The irradiance color value is then output to surface color modulator unit 1430.

Dot product unit 1420 computes the dot product of the vector I_e output by look-up unit 1410 and the surface normal in emitter space N_e. This dot product calculation only requires 3 multiply operations and 2 add operations. Dot product unit 1420 then outputs the irradiance energy component E.

Surface color modulator unit 1430 modulates irradiance energy component E with the irradiance color output by look-up unit 1410 to obtain an irradiance color modulated irradiance energy component E_c as described with respect to step 1232. Irradiance color modulated irradiance energy component E_c is then used to modulate surface color in an operation that only requires 3

multiply operations. Surface color modulator unit 1430 outputs an shading color representative of incident point or area illumination which is then output as the final shaded color or combined with further lighting and coloring operations in a shader.

An optional unit (not shown) for computing light visibility information (or occlusion), as described above, can used to scale the irradiance energy component E output from dot product unit 1420 and/or to scale the shading color from result from surface color modulator unit 1430.

Thus, in this embodiment, the present invention can represent complex point and area light sources with color inexpensively with little additional hardware.

c. Third Approach: Special Case of Constant Color Sources

The first and second approaches described above apply to any color source. In the case of a constant color source, a third approach can be used which is even more efficient. In particular, as shown in routine 1500 in FIG. 15 to provide constant color illumination, routine 100 need only be modified such that the modulated surface color obtained in step 140 is further multiplied by the light source color (step 1550).

Similarly, to provide constant color illumination, routine 300 need only be modified such that after step 360, the fifth frame buffer image FB5 is further multiplied by the light source color to obtain a sixth frame buffer image FB6. In step 370, the sixth frame buffer image FB6 is then multiplied by the fourth frame buffer image FB4 to obtain the final frame buffer display image.

To provide constant color illumination using the hardware of FIG. 8, an additional light source color input is provided to surface color modulator unit 830. The modulated surface color obtained by surface color modulator unit 830 is further multiplied by the light source color. Compared to the system of FIG. 14, no additional memory 1412 or look-up operations to the irradiance color table in memory 1412 are needed.

d. Color Illumination with Multi-Resolutional Format

Finally, each of the above three approaches for color illumination can be used to provide color illumination in routine 500 based on multi-resolutional irradiance field tables as would be apparent to a person skilled in the relevant art(s) given this description. Namely, under the first approach, a multi-resolution set of irradiance color vector field tables is generated in step 510. In step 522, the irradiance color tables covering the point with finest resolution are selected. In step

524, irradiance color vectors are computed using the irradiance color tables selected in step 522. In step 530, at each shading point, a dot product operation is performed between each irradiance color vector (I_R, I_B, I_R) and a surface normal N_e. In step 540, surface color is then modulated based on the irradiance energy component as described with respect to step 1140.

Under the second approach, one set of multi-resolutional irradiance color tables are also generated in step 510. In step 522, an irradiance color table covering the point with finest resolution is selected. An irradiance color is computed as described with respect to step 1222. Between steps 530 and 540, the irradiance color is used to further modulate the irradiance energy component as described with respect to steps 1232 and 1240.

Under the third approach, in the special case of constant color source; routine 500 need only be modified such that the modulated surface color obtained in step 540 is further multiplied by the light source color.

Pseudocode

Pseudocode for an example of the second approach to color illumination described above is provided below.

Example 3 With Color

```
float dot (Vector V Normal N)
{
return (V.x * N.x) + (V.y * N.y)+ (V.z * N.z);
}
/*      transformation that is applied to the emitter */
Transformation world2emitter;
/*      multi-resolution representation of the
irradiance field */
VectorField irradiance-field;

Color shade_pixel (Point P, Normal N, Color surface_color)
{
        Vector irradiance_vector;
```

```
float modulate;

/*    transform the point being shaded to the
      coordinate space of the emitter*/
P =   world2emitter.transform(P);
N =   world2emitter.transform(N);

/*    use this point to look up the irradiance
      vector*/
irradiance_vector = irradiance_field.lookup(P);
Color light_color = irradiance_color_table.lookup(P);

/*    compute the dot product to find the incoming
      irradiance */
modulate = dot (irradiance_vector, N);

/*    use the result to modulate the surface's color */ return C * modulate * light_color;
}
```

16. Conclusion

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. It will be understood by those skilled in the relevant art(s) that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined in the appended claims. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What Is Claimed Is:

1. A method for providing color illumination in computer graphics shading, comprising the steps of:

storing at least one color irradiance vector field table, each color irradiance vector field table being representative of an irradiance field at a respective spectral wavelength for a scene to be rendered; and

lighting the scene based on color irradiance vectors in said irradiance vector field table.

2. The method of claim 1, wherein said storing step comprises storing a set of color irradiance vector field tables, each color irradiance vector field table being representative of an irradiance field at a respective spectral wavelength for a scene to be rendered; and wherein said lighting step, for each point being lit, comprises the steps of:

computing a set of color irradiance vectors for each point using the set of irradiance vector field tables; and

performing a vector operation between each color irradiance vector in said set of color irradiance vectors and a surface normal for the point to compute an irradiance energy component; whereby, the irradiance energy component can be used in determining a shade value for the point.

3. The method of claim 2, wherein said lighting step, for each point being lit, further comprises the step of modulating surface color of the point based on the irradiance energy component.

4. The method of claim 2, wherein said color irradiance vector computing step comprises for each color irradiance vector to be computed the steps of:

looking up color irradiance vectors for points near to a point of interest in a respective color irradiance vector field table; and

interpolating the color irradiance vectors found in said looking up step to compute the color irradiance vector at the respective spectral wavelength for the point of interest.

5. The method of claim 2, wherein said vector operation performing step comprises the steps of calculating and summing dot products between each color irradiance vector in said set of color irradiance vectors and a surface normal for the point.

6. The method of claim 1, further comprising, prior to said storing step, the steps of generating each color irradiance vector field table in emitter space, and transforming each point and corresponding surface normal to emitter space, prior to said lighting step.

7. The method of claim 1, wherein said storing step stores a three-dimensional color irradiance vector field table representative of a three-dimensional irradiance field at a respective spectral wavelength.

8. The method of claim 1, wherein said storing step stores a color irradiance vector field table representative of an irradiance field of at least one point color light source.

9. The method of claim 8, wherein said storing step stores a color irradiance vector field table representative of an irradiance field of at least one point light source defined by a luminaire file format and a spectral wavelength.

10. The method of claim 1, wherein said storing step stores a color irradiance vector field table representative of an irradiance field of at least one area color light source.

11. The method of claim 1, further comprising the step of generating each color irradiance vector field table.

12. The method of claim 1, wherein said storing step stores a set of multi-resolutional color irradiance field tables covering progressively larger regions of a scene at progressively coarser resolutions.

13. The method of claim 1, wherein said storing step further includes the step of storing encoded visibility information that represents shadows in the irradiance field.

14. The method of claim 1, wherein said storing step further includes the step of storing encoded visibility information that represents participating media in the irradiance field.

15. The method of claim 2, further comprising the steps of:
determining visibility information from a shadow map; and
scaling said irradiance energy component based on said visibility information.

16. The method of claim 2, further comprising the steps of:
performing a back-projection calculation to determine visibility information; and
scaling said irradiance energy component based on said visibility information.

17. The method of claim 2, further comprising the steps of:
performing a shadow volume calculation to determine visibility information; and
scaling said irradiance energy component based on said visibility information.

18. The method of claim 1, wherein said lighting step lights a scene in an animation frame; whereby, color illumination in computer graphics animation is provided.

19. A method for providing illumination in computer graphics shading, comprising the steps of:
generating an irradiance vector field table representative of an irradiance field for a scene to be rendered; and
lighting the scene based on irradiance vectors in said irradiance vector field table.

20. The method of claim 19, wherein said lighting step, for each point being lit, comprises the steps of:
computing an irradiance vector for a point using the irradiance vector field table; and
performing a vector operation between the irradiance vector and a surface normal for the point to compute an irradiance energy component; whereby, the irradiance energy component can be used in determining a shade value for the point.

21. The method of claim 20, wherein said lighting step, for each point being lit, further comprises the step of modulating surface color of the point based on the irradiance energy component.

22. The method of claim 20, wherein said irradiance vector computing step comprises the steps of:

looking up irradiance vectors for points adjacent to a point of interest in the irradiance vector field table; and

interpolating the irradiance vectors found in said looking up step to compute the irradiance vector for the point of interest.

23. The method of claim 20, wherein said vector operation performing step comprises the step of calculating a dot product between the irradiance vector and the surface normal for the point.

24. The method of claim 20, wherein said generating step generates the irradiance vector field table in emitter space, and further comprising the step of transforming each point and corresponding surface normal to emitter space, prior to said lighting step.

25. The method of claim 19, wherein said generating step generates a three-dimensional irradiance vector field table representative of a three-dimensional irradiance field.

26. The method of claim 19, wherein said generating step generates an irradiance vector field table representative of an irradiance field of at least one point light source.

27. The method of claim 26, wherein said generating step generates an irradiance vector field table representative of an irradiance field of at least one point light source defined by a luminaire file format.

28. The method of claim 19, wherein said generating step generates an irradiance vector field table representative of an irradiance field of at least one area light source.

29. The method of claim 19, further comprising the step of storing said irradiance vector field table in memory.

30. The method of claim 19, wherein said generating step generates a set of multi-resolutional irradiance field tables covering progressively larger regions of a scene at progressively coarser resolutions.

31. The method of claim 19, wherein said generating step further comprises the step of encoding visibility information to represent shadows in the irradiance field.

32. The method of claim 19, wherein said generating step further comprises the step of encoding visibility information to represent participating media in the irradiance field.

33. The method of claim 20, further comprising the steps of:
determining visibility information from a shadow map; and
scaling said irradiance energy component based on said visibility information.

34. The method of claim 20, further comprising the steps of:
performing a back-projection calculation to determine visibility information; and
scaling said irradiance energy component based on said visibility information.

35. The method of claim 20, further comprising the steps of:
performing a shadow volume calculation to determine visibility information; and
scaling said irradiance energy component based on said visibility information.

36. A method for providing color illumination in computer graphics shading, comprising the steps of:

storing an irradiance vector field table representative of an irradiance field for a scene to be rendered;

storing an irradiance color table representative of color for a scene to be rendered; and

lighting the scene based on irradiance vectors in said irradiance vector field table and irradiance color values in said irradiance color table.

37. The method of claim 36, wherein said lighting step, for each point being lit, comprises the steps of:

computing an irradiance vector for a point using the irradiance vector field table;

computing an irradiance color for the point using the irradiance color table;

performing a vector operation between the irradiance vector and a surface normal for the point to compute an irradiance energy component; and

modulating the irradiance energy component with the computed irradiance color; whereby, the modulated irradiance energy component can be used in determining a shade value for the point.

38. The method of claim 37, wherein said lighting step, for each point being lit, further comprises the step of modulating surface color of the point based on the modulated irradiance energy component.

39. The method of claim 37, wherein said irradiance vector computing step comprises the steps of:

looking up irradiance vectors for points near to a point of interest in the irradiance vector field table; and

interpolating the irradiance vectors found in said looking up step to compute the irradiance vector for the point of interest.

40. The method of claim 37, wherein said vector operation performing step comprises the step of calculating a dot product between the irradiance vector and a surface normal for the point.

41. The method of claim 37, wherein said irradiance color computing step comprises the steps of:

looking up irradiance colors for points near to a point of interest in the irradiance color table; and

interpolating the irradiance colors found in said looking up step to compute the irradiance color for the point of interest.

42. The method of claim 36, wherein said storing steps comprise storing a set of multi-resolutional irradiance vector field tables and a set of multi-resolutional irradiance color tables covering progressively larger regions of a scene at progressively coarser resolutions.

43. The method of claim 42, wherein said lighting step includes selecting an irradiance vector field table and an irradiance color table covering a point with finest resolution.

44. The method of claim 36, further comprising, prior to said storing steps, the step of generating the irradiance vector field table and the irradiance color table.

45. A method for shading a computer graphics image to represent color illumination from a source, comprising the steps of:

storing an irradiance vector field table as first texture;

storing an irradiance color table as a second texture;

rendering a first image using the first texture to color pixels;

rendering a second image having surface normals encoded in red, green, and blue (RGB) color components;

multiplying the first and second images to obtain a third image;

accumulating components in said third image to obtain a fourth image;

rendering a scene with at least one of color and texture to obtain a fifth image;

multiplying the fourth and fifth images to obtain a frame buffer display image;

rendering a scene with second texture to obtain a sixth image; and

multiplying the sixth image and the frame buffer display image to obtain a final frame buffer display image.

46. A method for providing illumination of constant color in computer graphics shading, comprising the steps of:

storing an irradiance vector field table representative of an irradiance field for a scene to be rendered;

computing an irradiance vector for a point being lit using the irradiance vector field table;

performing a vector operation between the irradiance vector and a surface normal for the point to compute an irradiance energy component;

modulating a surface color of the point based on the irradiance energy component to obtain a modulated surface color value; and

multiplying the modulated surface color by a light source color value.

47. A method for shading a computer graphics image to represent illumination from a source of constant color, comprising the steps of:

storing an irradiance vector field table as irradiance texture;

rendering a first image using the irradiance texture to color pixels;

rendering a second image having surface normals encoded in red, green, and blue (RGB) color components;

multiplying the first and second images to obtain a third image;

accumulating components in said third image to obtain a fourth image;

rendering a scene with at least one of color and texture to obtain a fifth image;

multiplying the fifth image by a light source color value to obtain a sixth frame buffer image; and

multiplying the fourth and sixth images to obtain a final frame buffer display image.

48. A method for shading a computer graphics image to represent illumination from a source, comprising the steps of:

generating an irradiance vector field table;

rendering a first image using the irradiance vector field table to color pixels;

rendering a second image having surface normals encoded in red, green, and blue (RGB) color components;

multiplying the first and second images to obtain a third image;

accumulating components in said third image to obtain a fourth image;

rendering a scene with at least one of color and texture to obtain a fifth image; and

multiplying the fourth and fifth images to obtain a final display image.

49. A method for providing illumination in computer graphics animation, comprising the steps of:

generating an irradiance vector field table representative of an irradiance field for a scene to be rendered; and

lighting the scene in an animation frame based on irradiance vectors in said irradiance vector field table.

50. A system for shading a point of a computer graphics image to represent illumination from a source, comprising:

a memory that stores an irradiance vector field table representative of an irradiance field for the source illumination;

a look-up unit that looks up and interpolates irradiance vectors for selected points in the irradiance vector field table to compute an interpolated irradiance vector; and

a dot product unit that calculates a dot product between said interpolated irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component; whereby, the irradiance energy component can be used in determining a final shading color value for the point.

51. The system of claim 50, wherein said memory comprises a texture memory for storing a three-dimensional irradiance vector field table representative of a three-dimensional irradiance field.

52. The system of claim 50, wherein the source illumination comprises at least one point or area light source, and said memory stores an irradiance vector field table representative of the irradiance field of the at least one point or area light source.

53. A system for shading a point of a computer graphics image to represent illumination from a source of constant color, comprising:

a memory that stores an irradiance vector field table representative of an irradiance field for the source illumination;

a look-up unit that looks up and interpolates irradiance vectors for selected points in the irradiance vector field table to compute an interpolated irradiance vector;

a dot product unit that calculates a dot product between said interpolated irradiance vector and a surface normal for the point being shaded and outputs an irradiance energy component;

a surface color modulator that modulates the surface color of the point based on irradiance energy component; and

a multiplier unit that multiplies the modulated surface color by a light source color value for the point.

54. A system for providing color illumination in computer graphics shading, comprising:

a memory for storing at least one color irradiance vector field table, each color irradiance vector field table being representative of an irradiance field at a respective spectral wavelength for a scene to be rendered; and

lighting means for lighting the scene based on color irradiance vectors in said irradiance vector field table.

55. The system of claim 54, wherein said memory stores a set of color irradiance vector field tables, each color irradiance vector field table being representative of an irradiance field at a respective spectral wavelength for a scene to be rendered; and wherein said lighting means, for each point being lit, comprises:

computing means for computing a set of color irradiance vectors for each point using the set of irradiance vector field tables; and

a vector operation performing means for performing a vector operation between each color irradiance vector in said set of color irradiance vectors and a surface normal for the point to compute an irradiance energy component; whereby, the irradiance energy component can be used in determining a shade value for the point.

56. The system of claim 55, wherein said lighting means, for each point being lit, further comprises means for modulating surface color of the point based on the irradiance energy component.

57. The system of claim 55, wherein said color irradiance vector computing means comprises:

look-up means for looking up color irradiance vectors for points adjacent to a point of interest in a respective color irradiance vector field table; and

interpolator means for interpolating the color irradiance vectors output by said looking up means to obtain the color irradiance vector at the respective spectral wavelength for the point of interest; and

wherein said vector operation performing means comprises dot product calculating means for calculating dot products between each color irradiance vector in said set of color irradiance vectors and a surface normal for the point.

58. A system for providing color illumination in computer graphics shading, comprising:

a memory for storing a set of color irradiance vector field tables, each color irradiance vector field table being representative of an irradiance field at a respective spectral wavelength for a scene to be rendered;

a look-up unit that looks up color irradiance vectors for points adjacent to a point of interest in a respective color irradiance vector field table;

an interpolator unit that interpolates the color irradiance vectors output by said looking up unit to obtain the color irradiance vector at the respective spectral wavelength for the point of interest; and

a dot product unit that calculates dot products between each color irradiance vector in said set of color irradiance vectors and a surface normal for the point to compute an irradiance energy component; whereby, the irradiance energy component can be used in determining a shade value for the point.

59. A system for providing color illumination in computer graphics shading, comprising:

a first memory location for storing an irradiance vector field table representative of an irradiance field for a scene to be rendered;

a second memory location for storing an irradiance color table representative of color illumination for a scene to be rendered; and

lighting means for lighting the scene based on color irradiance vectors in said irradiance vector field table.

60. The system of claim 59, wherein said lighting means, for each point being lit, comprises:

computing means for computing an irradiance vector for each point using the irradiance vector field table;

computing means for computing an irradiance color for each point using the irradiance color table; and

a vector operation performing means for performing a vector operation between the irradiance vector and a surface normal for the point to compute an irradiance energy component; and

modulating means for modulating the irradiance energy component with the interpolated irradiance color to obtain a modulated irradiance energy component that can be used to determine a shade value for the point of interest to be shaded.

61. A system for providing color illumination in computer graphics shading, comprising:

a first memory location for storing an irradiance vector field table representative of an irradiance field for a scene to be rendered;

a second memory location for storing an irradiance color table representative of color illumination for a scene to be rendered;

a look-up unit that looks up irradiance vectors for points adjacent to a point of interest in the irradiance vector field table and that looks up irradiance colors for points adjacent to a point of interest in the irradiance color table;

an interpolator unit that interpolates the irradiance vectors output by said look-up unit to obtain an interpolated irradiance vector for the point of interest and that interpolates the irradiance colors output by said look-up unit to obtain an interpolated irradiance color for the point of interest;

a dot product unit that calculates a dot product between the interpolated irradiance vector and a surface normal for the point to compute an irradiance energy component; and

a modulator unit that modulates the irradiance energy component with the interpolated irradiance color to obtain a modulated irradiance energy component that can be used to determine a shade value for the point of interest to be shaded.

62. A computer program product comprising a computer useable medium having computer program logic recorded thereon for enabling a graphics processor in a computer system to provide illumination in computer graphics shading, said computer program logic comprising:

means for enabling the graphics processor to generate an irradiance vector field table representative of an irradiance field for a scene to be rendered;

means for enabling the graphics processor to light the scene based on irradiance vectors in said irradiance vector field table.

63. The computer program product of claim 62, wherein said lighting enabling means comprises:

means for enabling the graphics processor to compute an irradiance vector for a point using said irradiance vector field table and to perform a vector operation between said irradiance vector and a surface normal for said point to compute an irradiance energy component; whereby, the irradiance energy component can be used in determining a shade value for the point.

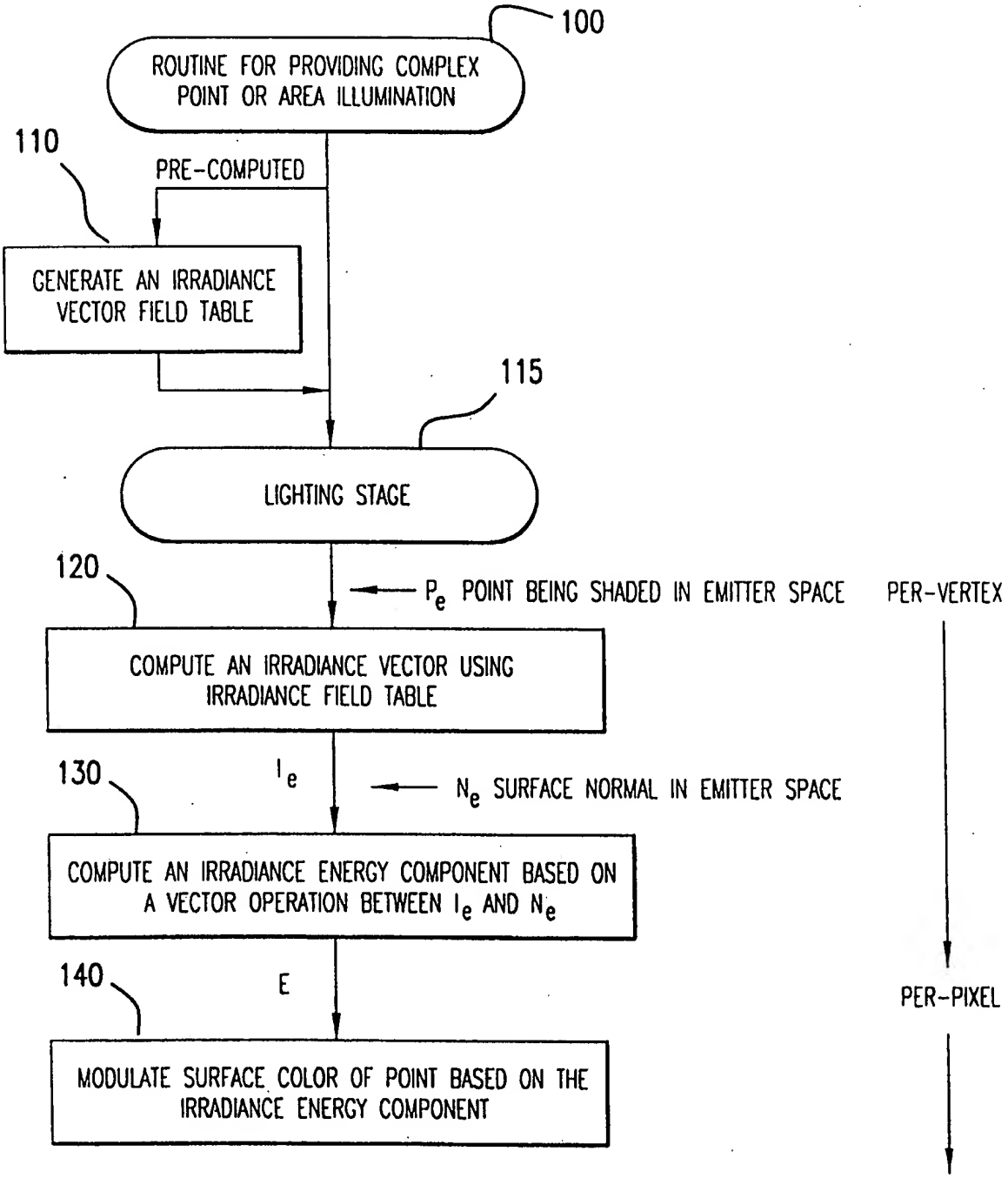

100
ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION
110
PRE-COMPUTED
GENERATE AN IRRADIANCE VECTOR FIELD TABLE
115
LIGHTING STAGE
120
P_e POINT BEING SHADED IN EMITTER SPACE
PER-VERTEX
COMPUTE AN IRRADIANCE VECTOR USING IRRADIANCE FIELD TABLE
130
I_e
N_e SURFACE NORMAL IN EMITTER SPACE
COMPUTE AN IRRADIANCE ENERGY COMPONENT BASED ON A VECTOR OPERATION BETWEEN I_e AND N_e
140
E
PER-PIXEL
MODULATE SURFACE COLOR OF POINT BASED ON THE IRRADIANCE ENERGY COMPONENT

FIG.1

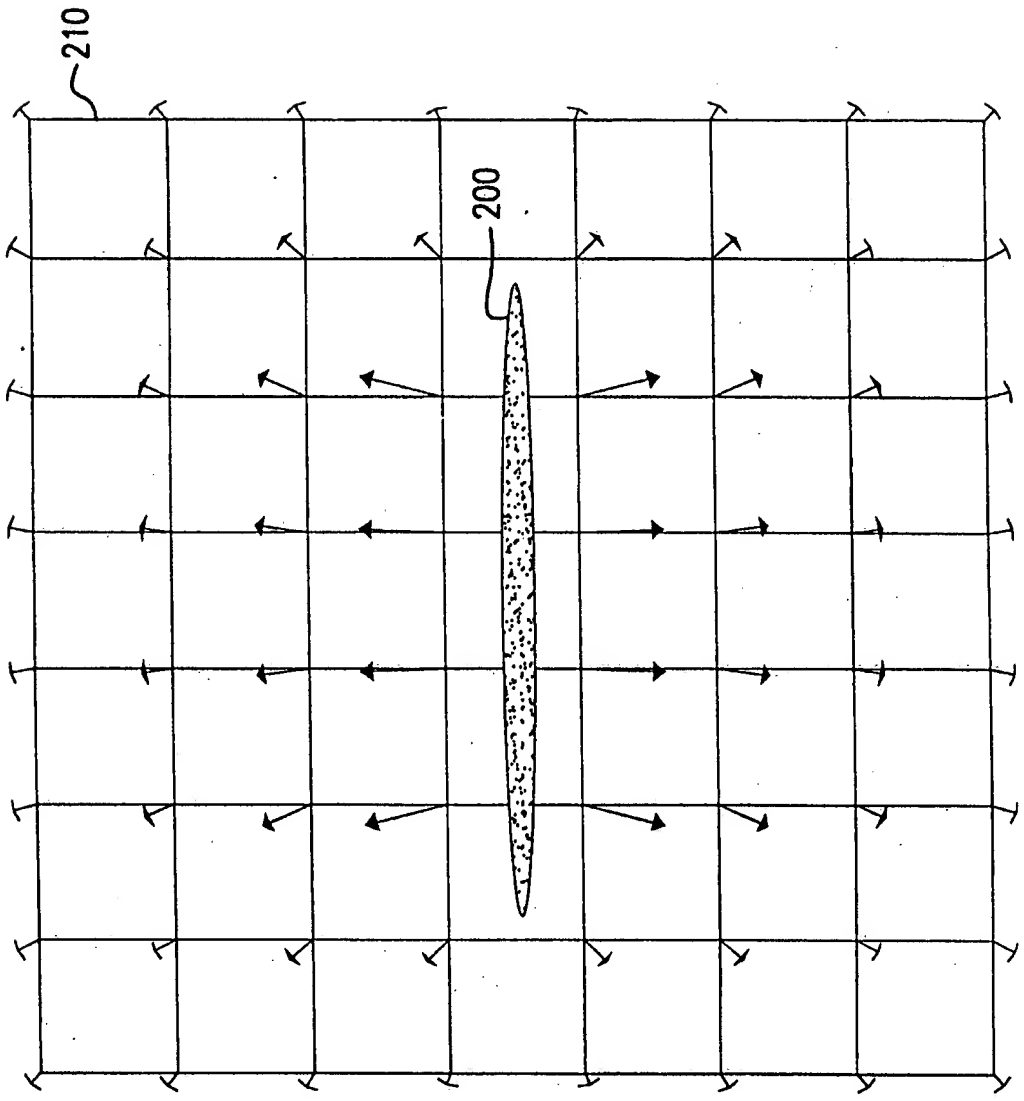


FIG.2A

POSITION	IRRADIANCE VECTOR
(-2.000, 0.0, -2.000)	(-0.058, 0.0, -0.066)
(-2.000, 0.0, -1.429)	(-0.077, 0.0, -0.065)
(-2.000, 0.0, -0.857)	(-0.085, 0.0, -0.046)
(-2.000, 0.0, -0.286)	(-0.044, 0.0, -0.008)
(-2.000, 0.0, 0.286)	(-0.044, 0.0, 0.008)
(-2.000, 0.0, 0.857)	(-0.085, 0.0, 0.046)
(-2.000, 0.0, 1.429)	(-0.077, 0.0, 0.065)
(-2.000, 0.0, 2.000)	(-0.058, 0.0, 0.066)
(-1.429, 0.0, -2.000)	(-0.066, 0.0, -0.108)
(-1.429, 0.0, -1.429)	(-0.105, 0.0, -0.133)
(-1.429, 0.0, -0.857)	(-0.159, 0.0, -0.140)
(-1.429, 0.0, -0.286)	(-0.148, 0.0, -0.054)
(-1.429, 0.0, 0.286)	(-0.148, 0.0, 0.054)
(-1.429, 0.0, 0.857)	(-0.159, 0.0, 0.140)
(-1.429, 0.0, 1.429)	(-0.105, 0.0, 0.133)
(-1.429, 0.0, 2.000)	(-0.066, 0.0, 0.108)
(-0.857, 0.0, -2.000)	(-0.056, 0.0, -0.159)
(-0.857, 0.0, -1.429)	(-0.102, 0.0, -0236)
(-0.857, 0.0, -0.857)	(-0.195, 0.0, -0.368)
(-0.857, 0.0, -0.286)	(-0.338, 0.0, -0.653)
(-0.857, 0.0, 0.286)	(-0.338, 0.0, 0.653)
(-0.857, 0.0, 0.857)	(-0.195, 0.0, 0.368)
(-0.857, 0.0, 1.429)	(-0.102, 0.0, 0.236)
(-0.857, 0.0, 2.000)	(-0.056, 0.0, 0.159)
(-0.286, .0., -2.000)	(-0.022, 0.0, -0.195)
(-0.286, 0.0, -1.429)	(-0.043, 0.0, -0.317)
(-0.286, 0.0, -0.857)	(-0.080, 0.0, -0.553)
(-0.286, 0.0, -0.286)	(-0.074, 0.0, -0.913)
(-0.286, 0.0, 0.286)	(-0.074, 0.0, 0.913)
(-0.286, 0.0, 0.857)	(-0.080, 0.0, 0.553)
(-0.286, 0.0, 1.429)	(-0.043, 0.0, 0.317)
(-0.286, 0.0, 2.000)	(-0.022, 0.0, 0.195)
(0.286, 0.0, -2.000)	(0.022, 0.0, -0.195)
(0.286, 0.0, -1.429)	(0.043, 0.0, -0.317)
(0.286, 0.0, -0.857)	(0.080, 0.0, -0.553)
(0.286, 0.0, -0.286)	(0.074, 0.0, -0.913)
(0.286, 0.0, 0.286)	(0.074, 0.0, 0.913)
(0.286, 0.0, 0.857)	(0.080, 0.0, 0.553)
(0.286, 0.0, 1.429)	(0.043, 0.0, 0.317)
(0.286, 0.0, 2.000)	(0.022, 0.0, 0.195)
(0.857, 0.0, -2.000)	(0.056, 0.0, -0.159)

FIG.2B

(0.857, 0.0, -1.429)	(0.102, 0.0, -0.236)
(0.857, 0.0, -0.857)	(0.195, 0.0, -0.368)
(0.857, 0.0, -0.286)	(0.338, 0.0, -0.653)
(0.857, 0.0, 0.286)	(0.338, 0.0, 0.653)
(0.857, 0.0, 0.857)	(0.195, 0.0, 0.368)
(0.857, 0.0, 1.1429)	(0.102, 0.0, 0.236)
(0.857, 0.0, 2.000)	(0.056, 0.0, 0.159)
(1.429, 0.0, -2.000)	(0.066, 0.0, -0.108)
(1.429, 0.0, -1.429)	(0.105, 0.0, -0.133)
(1.429, 0.0, -0.857)	(0.159, 0.0, -0.140)
(1.429, 0.0, -0.286)	(0.148, 0.0, -0.054)
(1.429, 0.0, 0.286)	(0.148, 0.0, 0.054)
(1.429, 0.0, 0.857)	(0.159, 0.0, 0.140)
(1.429, 0.0, 1.429)	(0.105, 0.0, 0.133)
(1.429, 0.0, 2.000)	(0.066, 0.0, 0.108)
(2.000, 0.0, -2.000)	(0.058, 0.0, -0.066)
(2.000, 0.0, -1.429)	(0.077, 0.0, -0.065)
(2.000, 0.0, -00857)	(0.085, 0.0, -0.046)
(2.000, 0.0, -0.286)	(0.044, 0.0, -0.008)
(2.000, 0.0, -0.286)	(0.044, 0.0, 0.008)
(2.000, 0.0, 0.857)	(0.085, 0.0, 0.046)
(2.000, 0.0, 1.429)	(0.077, 0.0, 0.065)
(2.000, 0.0, 2.000)	(0.058, 0.0, 0.066)

FIG.2B-1

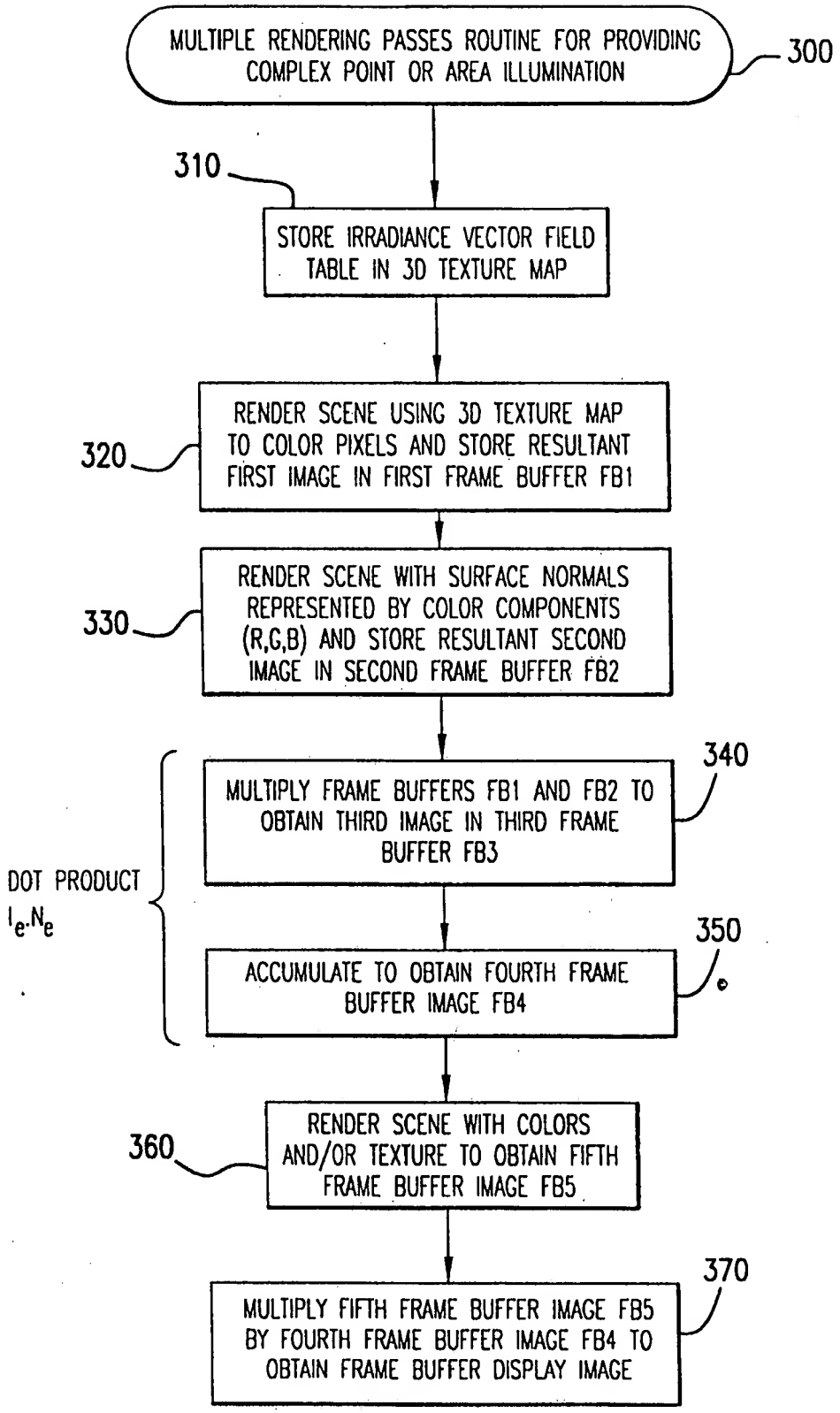

MULTIPLE RENDERING PASSES ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION
300
310
STORE IRRADIANCE VECTOR FIELD TABLE IN 3D TEXTURE MAP
320
RENDER SCENE USING 3D TEXTURE MAP TO COLOR PIXELS AND STORE RESULTANT FIRST IMAGE IN FIRST FRAME BUFFER FB1
330
RENDER SCENE WITH SURFACE NORMALS REPRESENTED BY COLOR COMPONENTS (R,G,B) AND STORE RESULTANT SECOND IMAGE IN SECOND FRAME BUFFER FB2
340
MULTIPLY FRAME BUFFERS FB1 AND FB2 TO OBTAIN THIRD IMAGE IN THIRD FRAME BUFFER FB3
DOT PRODUCT
$I_e \cdot N_e$
350
ACCUMULATE TO OBTAIN FOURTH FRAME BUFFER IMAGE FB4
360
RENDER SCENE WITH COLORS AND/OR TEXTURE TO OBTAIN FIFTH FRAME BUFFER IMAGE FB5
370
MULTIPLY FIFTH FRAME BUFFER IMAGE FB5 BY FOURTH FRAME BUFFER IMAGE FB4 TO OBTAIN FRAME BUFFER DISPLAY IMAGE

FIG.3

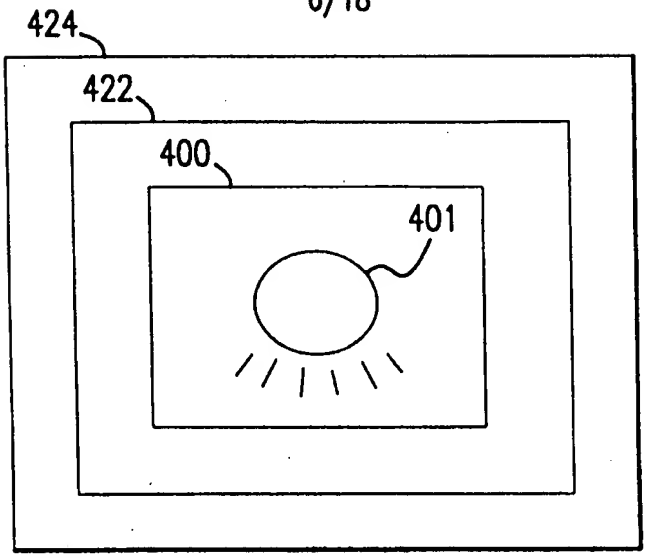
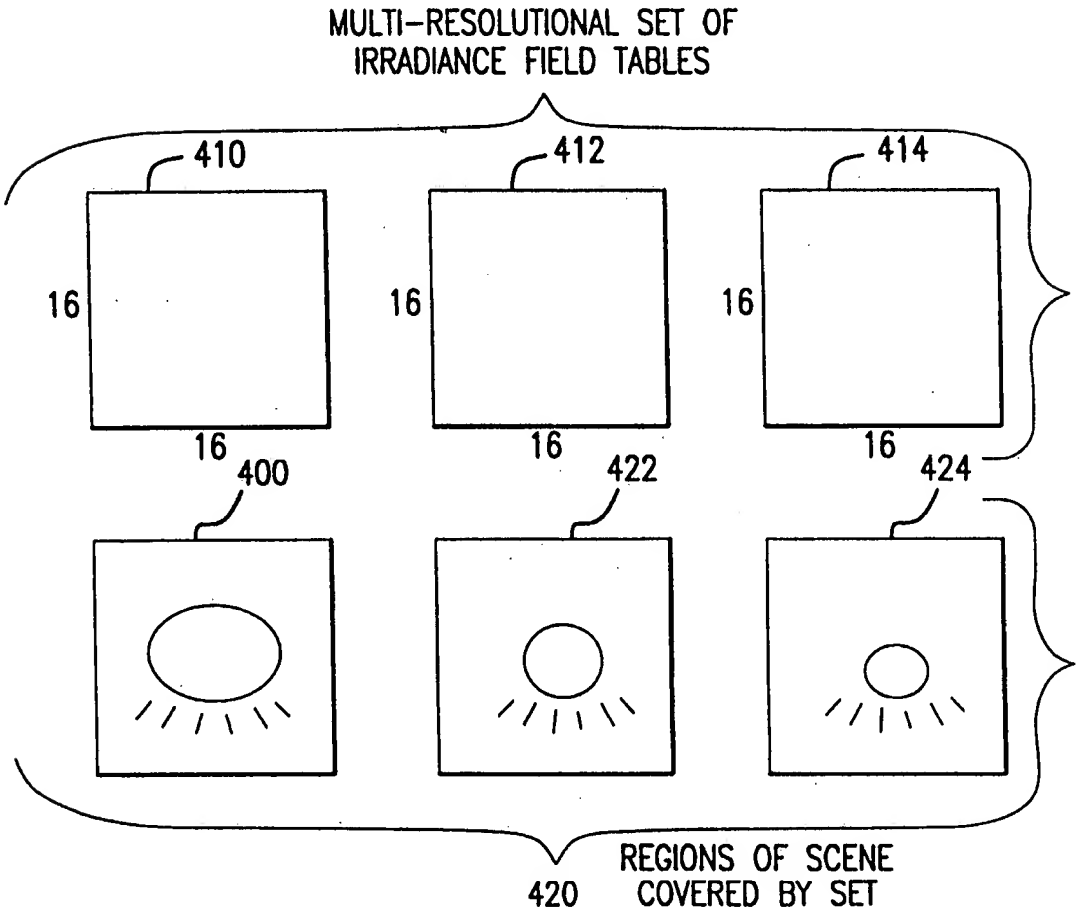
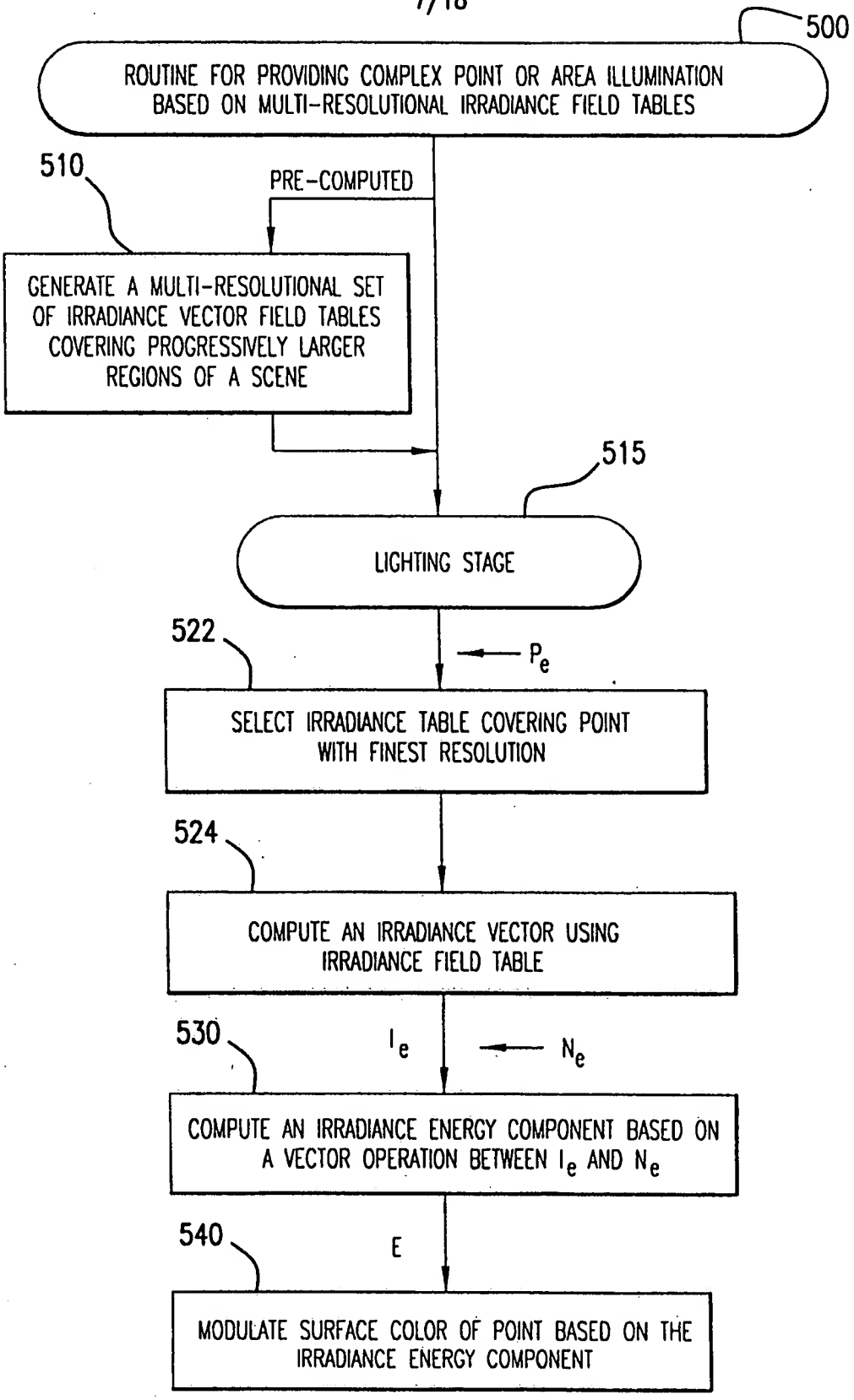

424
422
400
401

FIG.4A


MULTI-RESOLUTIONAL SET OF IRRADIANCE FIELD TABLES
410
412
414
16
16
16
16
16
16
400
422
424
REGIONS OF SCENE COVERED BY SET
420


500
ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION BASED ON MULTI-RESOLUTIONAL IRRADIANCE FIELD TABLES
510
PRE-COMPUTED
GENERATE A MULTI-RESOLUTIONAL SET OF IRRADIANCE VECTOR FIELD TABLES COVERING PROGRESSIVELY LARGER REGIONS OF A SCENE
515
LIGHTING STAGE
522
P_e
SELECT IRRADIANCE TABLE COVERING POINT WITH FINEST RESOLUTION
524
COMPUTE AN IRRADIANCE VECTOR USING IRRADIANCE FIELD TABLE
530
I_e
N_e
COMPUTE AN IRRADIANCE ENERGY COMPONENT BASED ON A VECTOR OPERATION BETWEEN I_e AND N_e
540
E
MODULATE SURFACE COLOR OF POINT BASED ON THE IRRADIANCE ENERGY COMPONENT

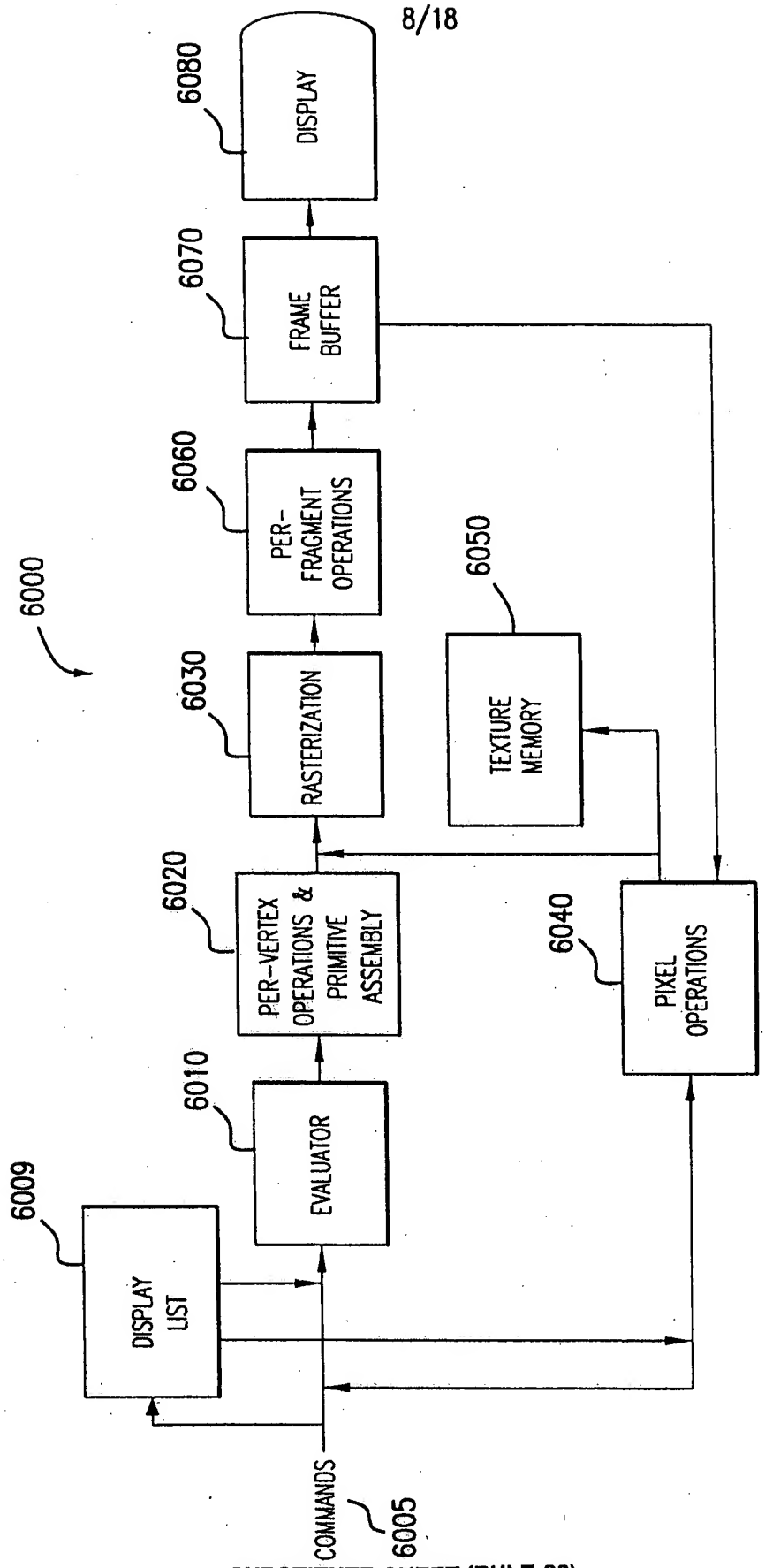

6000
6080
DISPLAY
6070
FRAME BUFFER
6060
PER-FRAGMENT OPERATIONS
6050
TEXTURE MEMORY
6030
RASTERIZATION
6020
PER-VERTEX OPERATIONS & PRIMITIVE ASSEMBLY
6040
PIXEL OPERATIONS
6010
EVALUATOR
6009
DISPLAY LIST
COMMANDS
6005

FIG.6

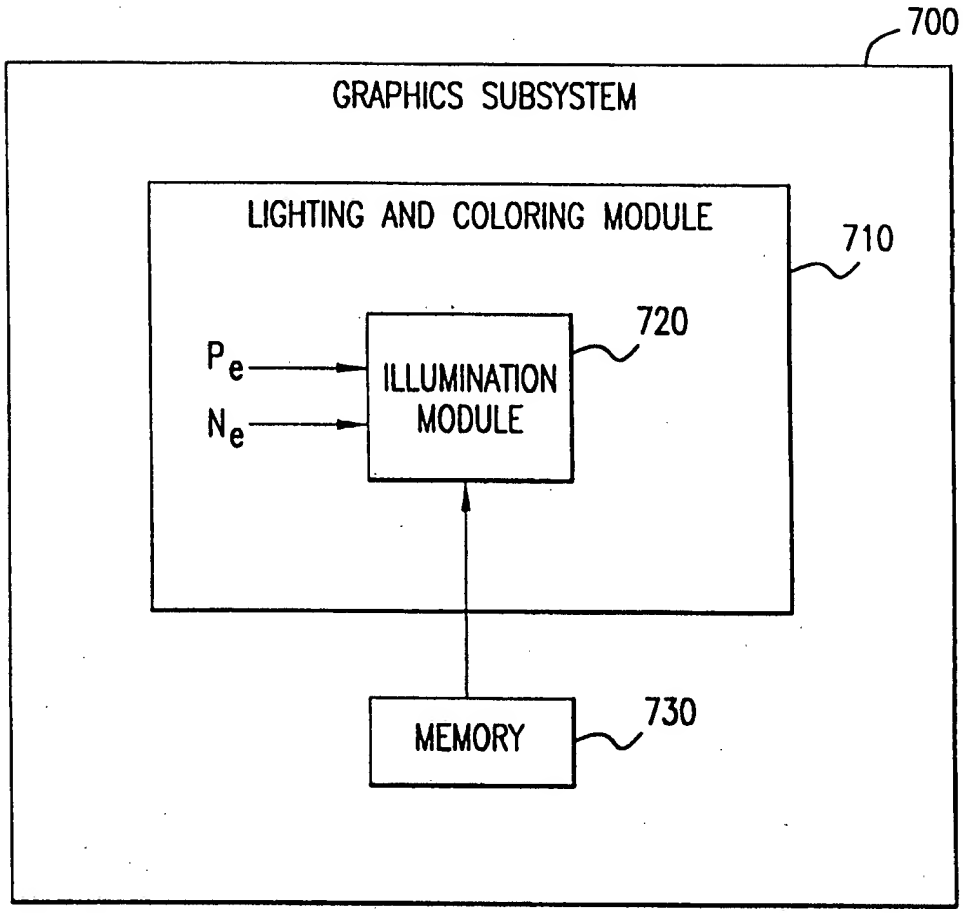

700
GRAPHICS SUBSYSTEM
LIGHTING AND COLORING MODULE
710
720
P_e
N_e
ILLUMINATION MODULE
MEMORY
730

FIG.7

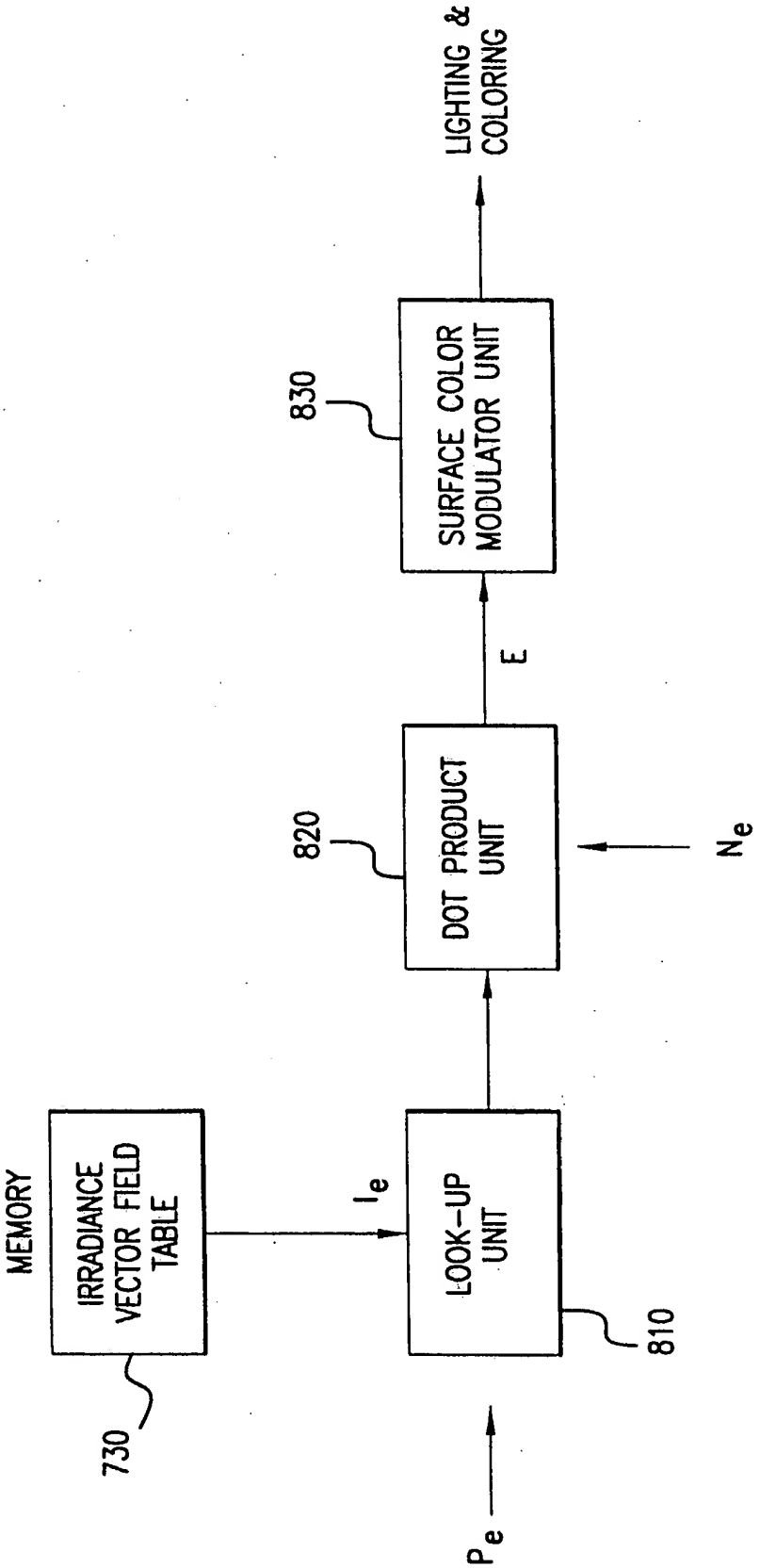

LIGHTING & COLORING
830
SURFACE COLOR MODULATOR UNIT
E
820
DOT PRODUCT UNIT
N_e
MEMORY
IRRADIANCE VECTOR FIELD TABLE
I_e
LOOK-UP UNIT
810
730
P_e

FIG.8

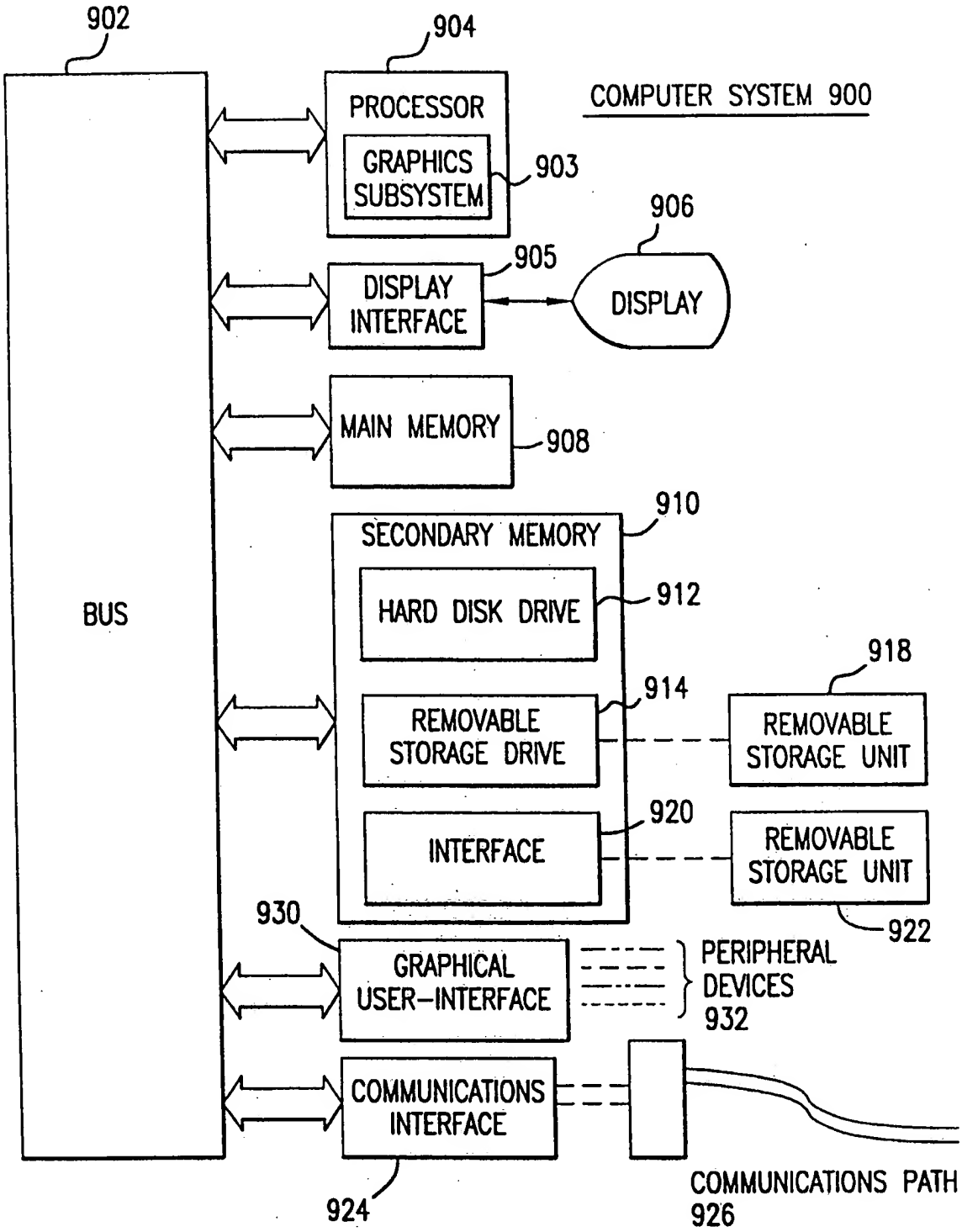

902
904
PROCESSOR
GRAPHICS SUBSYSTEM
903
COMPUTER SYSTEM 900
906
905
DISPLAY INTERFACE
DISPLAY
MAIN MEMORY
908
910
SECONDARY MEMORY
HARD DISK DRIVE
912
BUS
918
914
REMOVABLE STORAGE DRIVE
REMOVABLE STORAGE UNIT
920
INTERFACE
REMOVABLE STORAGE UNIT
922
930
GRAPHICAL USER-INTERFACE
PERIPHERAL DEVICES
932
COMMUNICATIONS INTERFACE
924
COMMUNICATIONS PATH
926

FIG.9

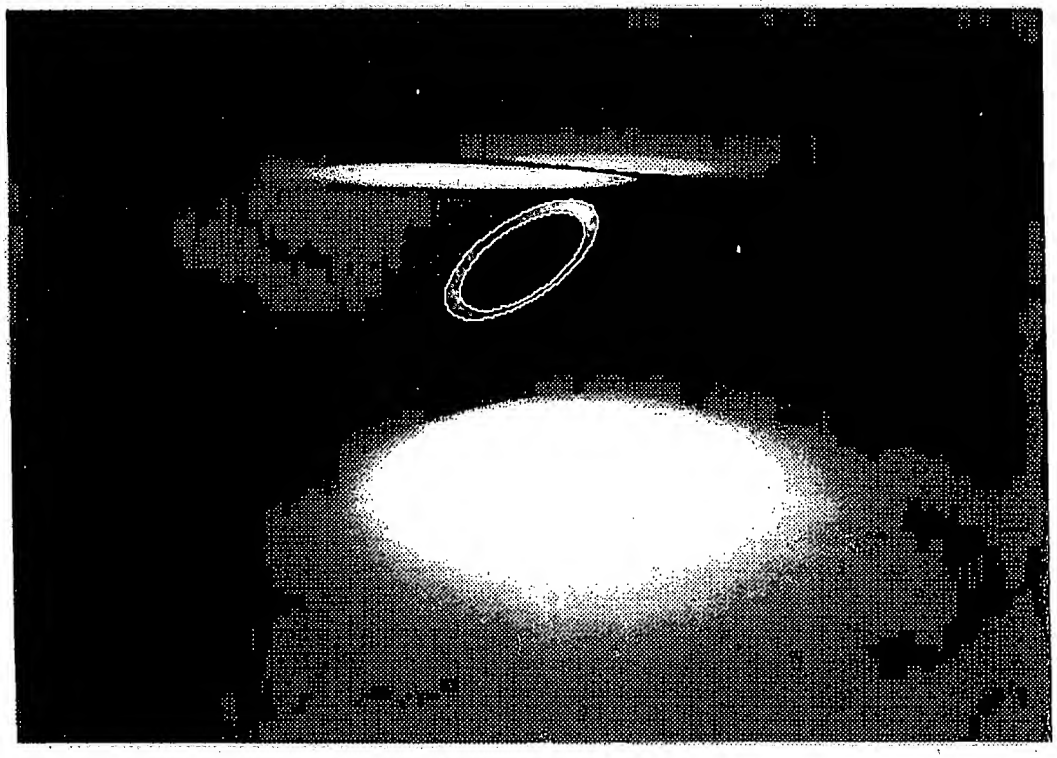


FIG.10A

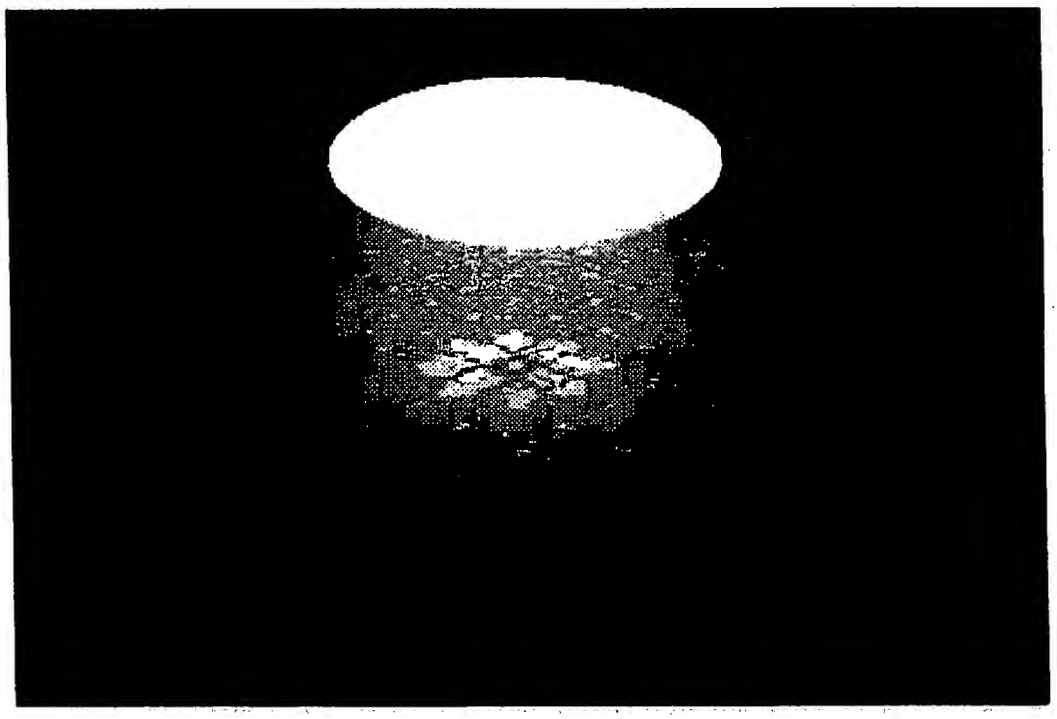


FIG. 10B

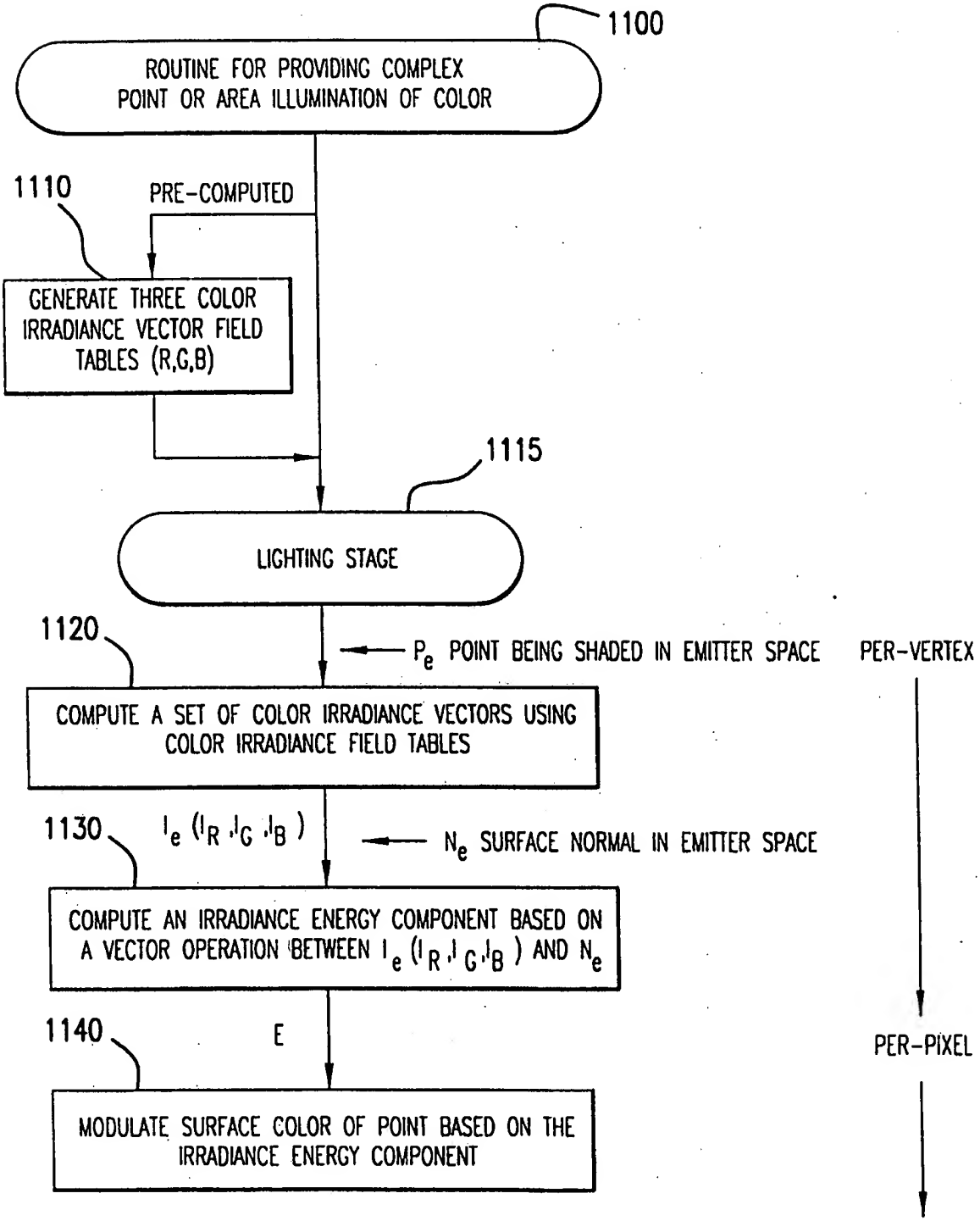

1100
ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION OF COLOR
1110
PRE-COMPUTED
GENERATE THREE COLOR IRRADIANCE VECTOR FIELD TABLES (R,G,B)
1115
LIGHTING STAGE
1120
P_e POINT BEING SHADED IN EMITTER SPACE
PER-VERTEX
COMPUTE A SET OF COLOR IRRADIANCE VECTORS USING COLOR IRRADIANCE FIELD TABLES
1130
$I_e (I_R, I_G, I_B)$
N_e SURFACE NORMAL IN EMITTER SPACE
COMPUTE AN IRRADIANCE ENERGY COMPONENT BASED ON A VECTOR OPERATION BETWEEN $I_e (I_R, I_G, I_B)$ AND N_e
1140
E
PER-PIXEL
MODULATE SURFACE COLOR OF POINT BASED ON THE IRRADIANCE ENERGY COMPONENT

FIG.11

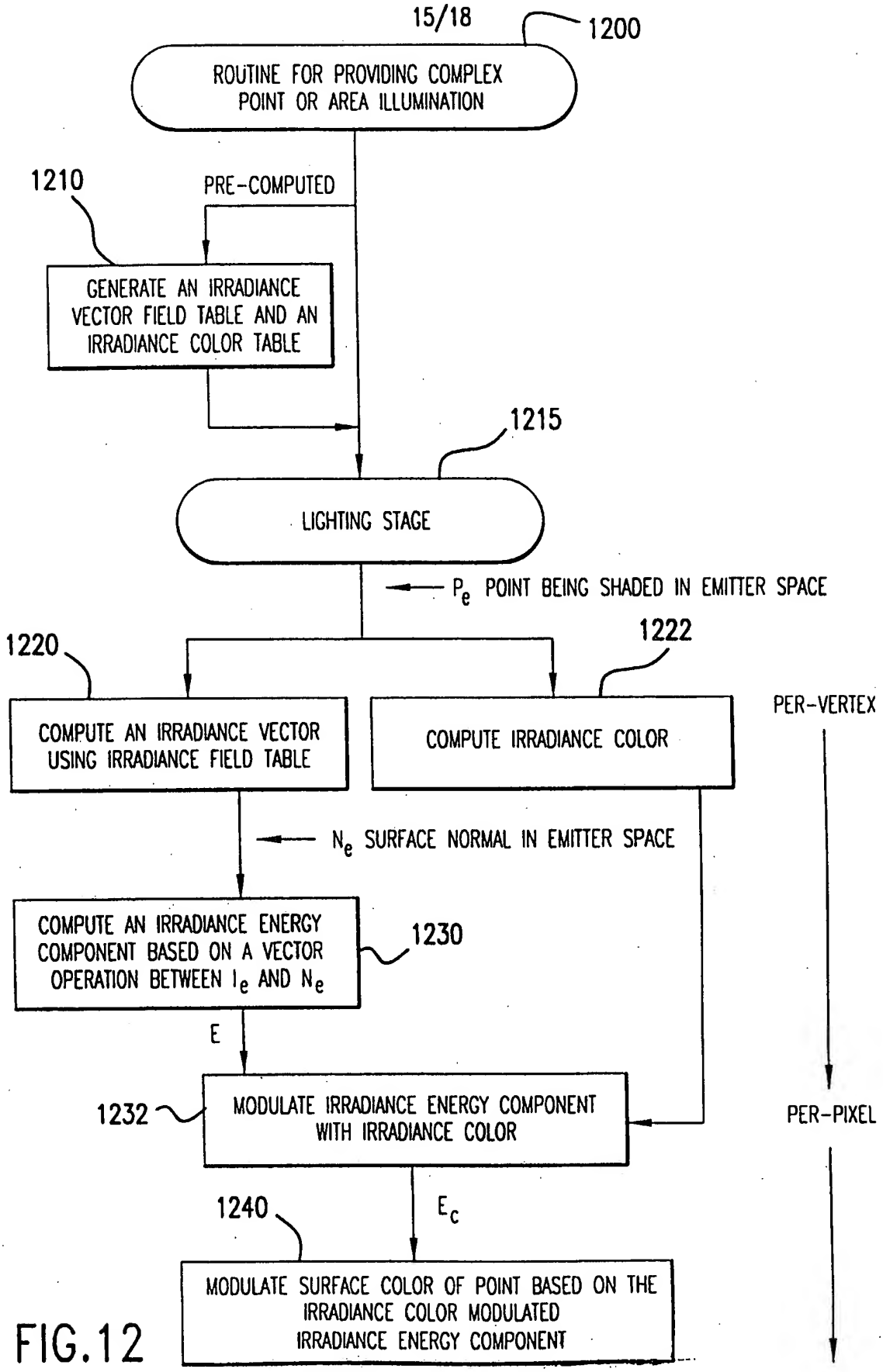

1200
ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION
1210
PRE-COMPUTED
GENERATE AN IRRADIANCE VECTOR FIELD TABLE AND AN IRRADIANCE COLOR TABLE
1215
LIGHTING STAGE
P_e POINT BEING SHADED IN EMITTER SPACE
1220
1222
COMPUTE AN IRRADIANCE VECTOR USING IRRADIANCE FIELD TABLE
COMPUTE IRRADIANCE COLOR
PER-VERTEX
N_e SURFACE NORMAL IN EMITTER SPACE
COMPUTE AN IRRADIANCE ENERGY COMPONENT BASED ON A VECTOR OPERATION BETWEEN I_e AND N_e
1230
E
1232
MODULATE IRRADIANCE ENERGY COMPONENT WITH IRRADIANCE COLOR
PER-PIXEL
1240
E_c
MODULATE SURFACE COLOR OF POINT BASED ON THE IRRADIANCE COLOR MODULATED IRRADIANCE ENERGY COMPONENT

FIG.12

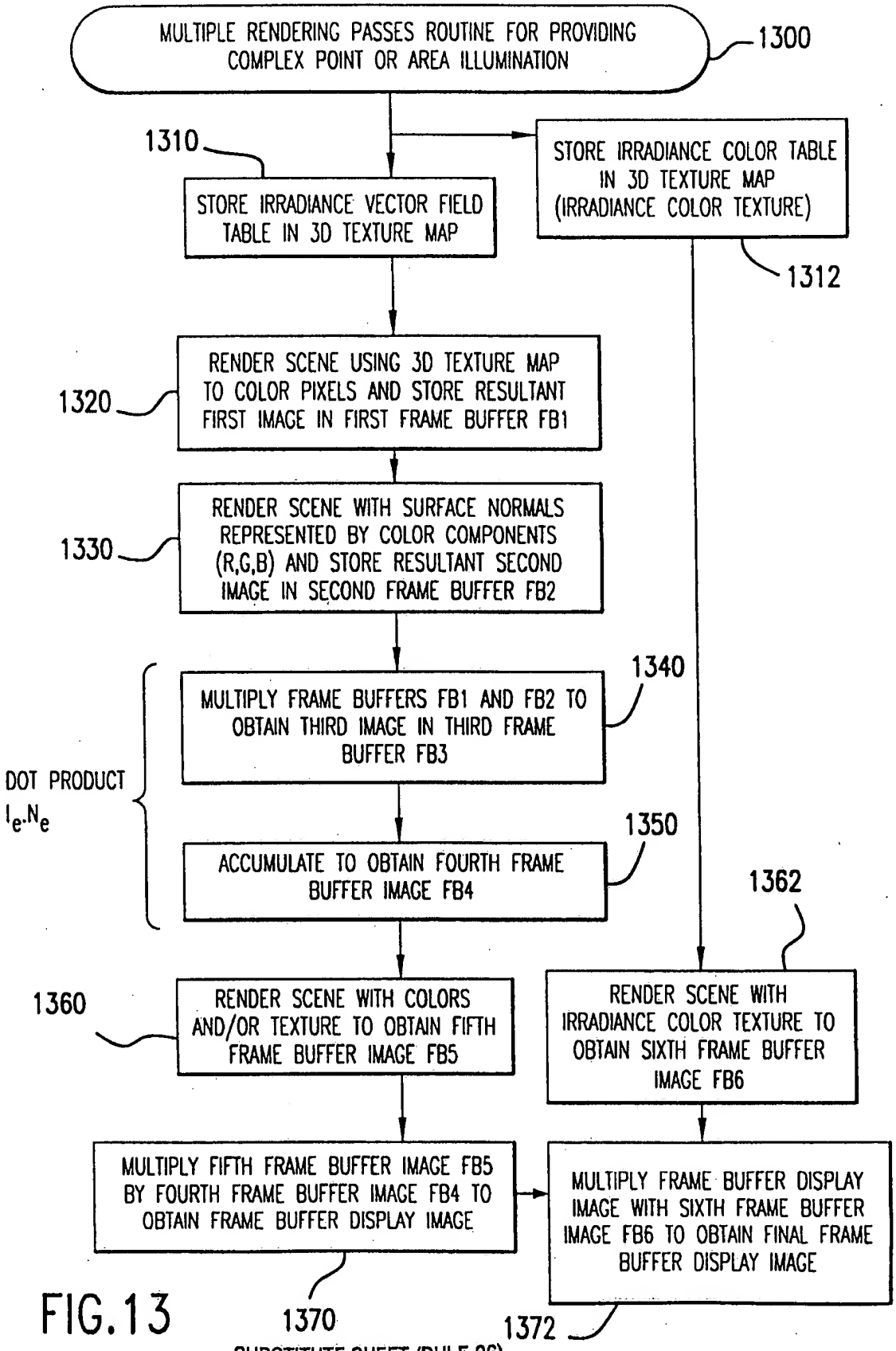

MULTIPLE RENDERING PASSES ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION
1300
1310
STORE IRRADIANCE VECTOR FIELD TABLE IN 3D TEXTURE MAP
STORE IRRADIANCE COLOR TABLE IN 3D TEXTURE MAP (IRRADIANCE COLOR TEXTURE)
1312
1320
RENDER SCENE USING 3D TEXTURE MAP TO COLOR PIXELS AND STORE RESULTANT FIRST IMAGE IN FIRST FRAME BUFFER FB1
1330
RENDER SCENE WITH SURFACE NORMALS REPRESENTED BY COLOR COMPONENTS (R,G,B) AND STORE RESULTANT SECOND IMAGE IN SECOND FRAME BUFFER FB2
1340
MULTIPLY FRAME BUFFERS FB1 AND FB2 TO OBTAIN THIRD IMAGE IN THIRD FRAME BUFFER FB3
DOT PRODUCT
$l_e \cdot N_e$
1350
ACCUMULATE TO OBTAIN FOURTH FRAME BUFFER IMAGE FB4
1362
1360
RENDER SCENE WITH COLORS AND/OR TEXTURE TO OBTAIN FIFTH FRAME BUFFER IMAGE FB5
RENDER SCENE WITH IRRADIANCE COLOR TEXTURE TO OBTAIN SIXTH FRAME BUFFER IMAGE FB6
MULTIPLY FIFTH FRAME BUFFER IMAGE FB5 BY FOURTH FRAME BUFFER IMAGE FB4 TO OBTAIN FRAME BUFFER DISPLAY IMAGE
MULTIPLY FRAME BUFFER DISPLAY IMAGE WITH SIXTH FRAME BUFFER IMAGE FB6 TO OBTAIN FINAL FRAME BUFFER DISPLAY IMAGE
1370
1372

FIG.13

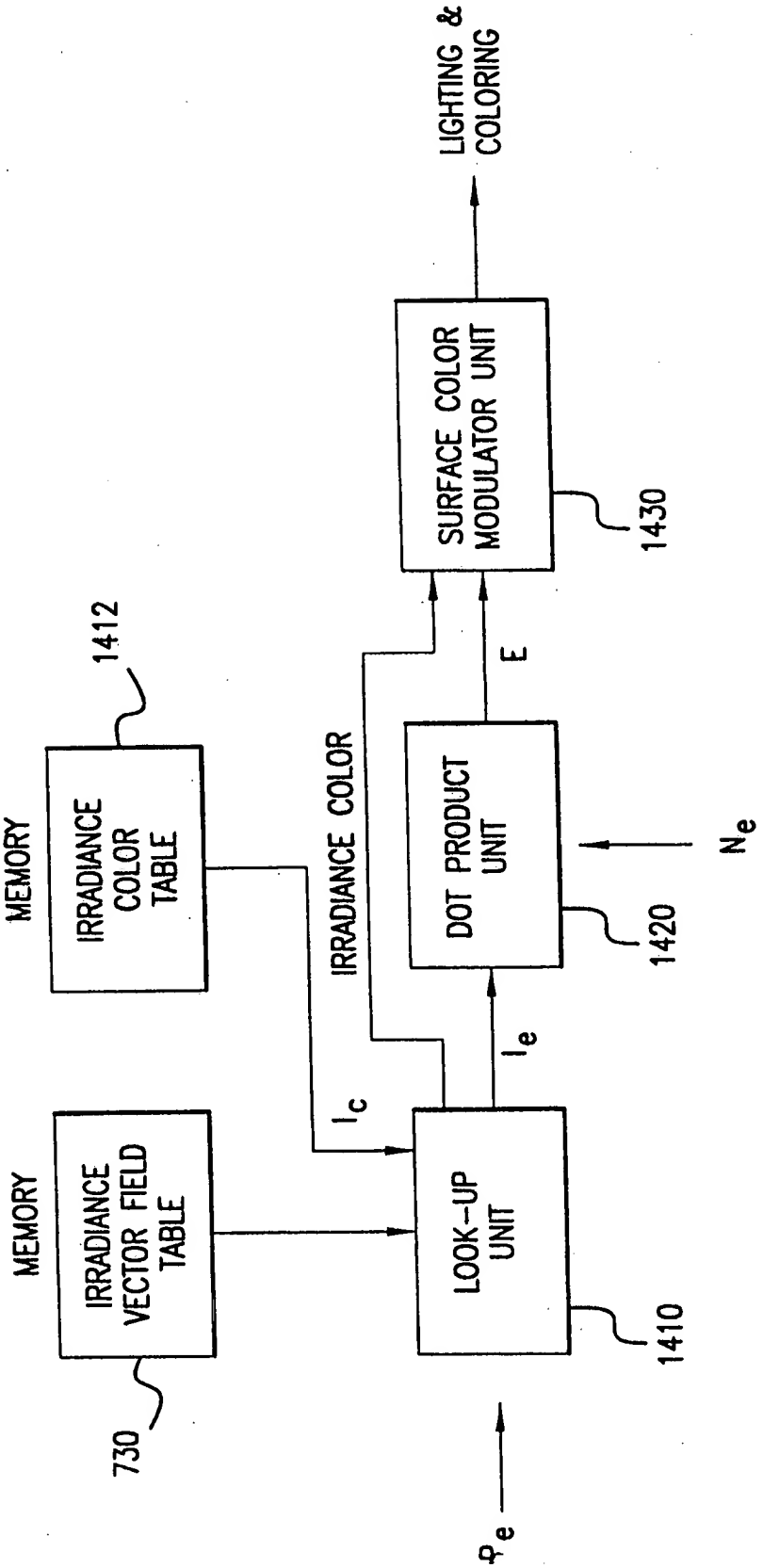

LIGHTING & COLORING
SURFACE COLOR MODULATOR UNIT
1430
E
DOT PRODUCT UNIT
1420
N_e
I_e
IRRADIANCE COLOR
MEMORY
IRRADIANCE COLOR TABLE
1412
I_c
MEMORY
IRRADIANCE VECTOR FIELD TABLE
730
LOOK-UP UNIT
1410
P_e

FIG.14

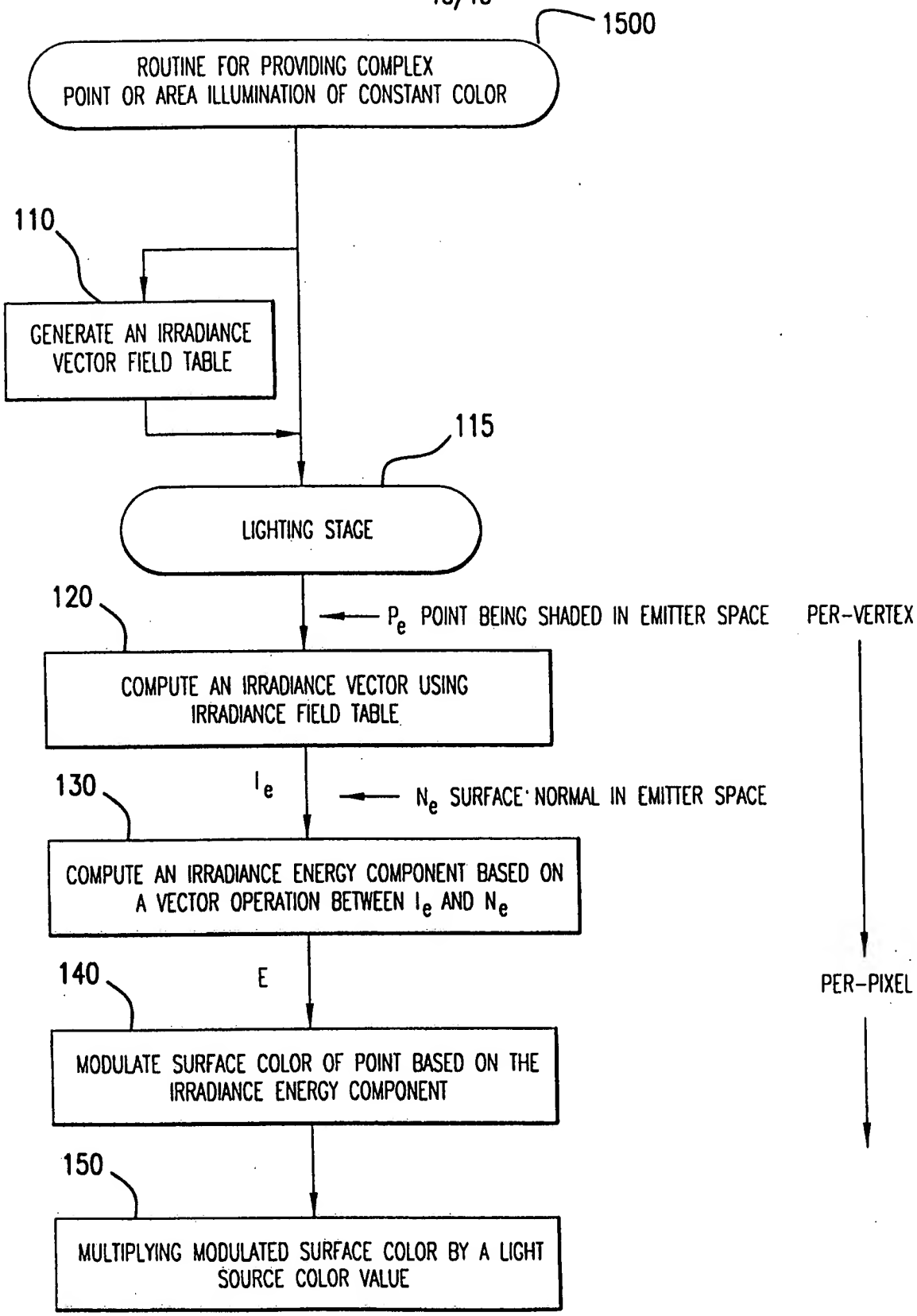

1500
ROUTINE FOR PROVIDING COMPLEX POINT OR AREA ILLUMINATION OF CONSTANT COLOR
110
GENERATE AN IRRADIANCE VECTOR FIELD TABLE
115
LIGHTING STAGE
120
P_e POINT BEING SHADED IN EMITTER SPACE
PER-VERTEX
COMPUTE AN IRRADIANCE VECTOR USING IRRADIANCE FIELD TABLE
130
I_e
N_e SURFACE NORMAL IN EMITTER SPACE
COMPUTE AN IRRADIANCE ENERGY COMPONENT BASED ON A VECTOR OPERATION BETWEEN I_e AND N_e
140
E
PER-PIXEL
MODULATE SURFACE COLOR OF POINT BASED ON THE IRRADIANCE ENERGY COMPONENT
150
MULTIPLYING MODULATED SURFACE COLOR BY A LIGHT SOURCE COLOR VALUE

FIG.15

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 98/20096

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06T15/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 709 231 A (SAKAIBARA TORU ET AL) 24 November 1987 see abstract see column 2, line 23 - line 59 see column 3, line 43 - column 4, line 44 see column 6, line 8 - line 17 see column 7, line 15 - line 22	1-63
X	US 5 268 996 A (STEINER WALTER R ET AL) 7 December 1993	62,63
A	see abstract see column 2, line 37 - line 49 see column 9, line 20 - line 38; figures 3,4	1-61

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 December 1998	23/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Gonzalez Ordonez, O

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter nal Application No
PCT/US 98/20096

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4709231 A	24-11-1987	JP 2025773 C	26-02-1996
		JP 7046391 B	17-05-1995
		JP 61070666 A	11-04-1986
US 5268996 A	07-12-1993	CA 2056457 A	21-06-1992
		FR 2670924 A	26-06-1992